THE UNIVERSITY OF ALBERTA

[illegible]

by

(C) [illegible]

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF [illegible]

EDMONTON, ALBERTA

[illegible]

THE UNIVERSITY OF ALBERTA

AN ON-LINE INFORMATION RETRIEVAL SYSTEM WITH AN APPLICATION TO WESTERN CANADIAN HISTORY

by

© Roger F. Halpin

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

SEPTEMBER, 1967

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled AN ON-LINE INFORMATION RETRIEVAL SYSTEM WITH AN APPLICATION TO WESTERN CANADIAN HISTORY submitted by Roger F. Halpin in partial fulfilment of the requirements for the degree of Master of Science.

Date September 26 1967

ABSTRACT

This thesis reviews problems in the information storage and retrieval cycle and describes the development of an experimental on-line storage and retrieval system (SARA) with a present data-base of documents in Western Canadian history. The review covers methods for converting information in documents to machine readable form, and the automatic analysis of this information to produce indexed documents, abstracts, and classifications; it describes briefly seven operational information storage and retrieval systems. SARA, which utilizes time-shared computing facilities and a new programming language called APL, is described in detail and evaluated.

ACKNOWLEDGEMENTS

I express my appreciation to Professor K.W. Smillie for the guidance given me in the preparation of this thesis, to Professor Doreen M. Heaps for her interest and assistance in this topic, and to Professor D.B. Scott, Head of the Department of Computing Science, for providing computing facilities and financial assistance while this research was being done. I also wish to thank the Department of History, and in particular Mr. J. Nicks, for the willing help given in the preparation of test material.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

GENERAL THESIS AND
PROBLEMS IN INFORMATION SYSTEMS

The phrase "information storage and retrieval" can take many meanings. For example, the local library and the office filing system can be considered information storage and retrieval systems. In the present thesis, the phrase will refer to systems which prepare and store documents and subsequently retrieve them (or their addresses) in response to requests. The systems may employ humans and machines such as computers.

The term "information" has been used in a multitude of ways. In this thesis, the term will refer to the characters, which, when ordered, form words. These characters may be printed or written on paper. The term will not imply that interpretation on the string of characters is done, or that meaning is derived from the string of characters.

Information storage includes the preparation of information which exists in document form and the storage of this information in some systematic manner suitable for subsequent retrieval. There are various definitions of information retrieval. Vickery (1965) states that "Retrieval is the selection of documentary information from a store, in

response to search questions.". Bourne (1963) differentiates between reference retrieval, document retrieval, fact retrieval and information retrieval. He defines document retrieval as a process which yields a complete copy of a document in response to a general search question, while information retrieval is a process which yields information to a request for information. Such a request might be: "What is the difference between a point-contact transistor and a junction transistor?". Kent (1962) gives a more general definition of machine literature searching, or information retrieval: "...the use of mechanized or other non-conventional tools in connection with any one or more unit operations" where a unit operation is "...a series of functions, or steps...".

Information storage and retrieval systems have been in operation since knowledge has been recorded. An example of such a system is a library. Information in the form of printed characters comprising documents is stored on shelves and indexed and catalogued for retrieval. In order to retrieve the information in the documents, the index cards are consulted and the proper documents are retrieved.

The volume of published material has been increasing at an exponential rate (see Bourne (1963)). Traditional techniques used to process the large volume of information cannot cope with the information explosion; new techniques

must be developed. Hence, machines, particulary computers, are being used to ease the effect of this explosion. This thesis concentrates on several specific applications of computers to information storage and retrieval although it acknowledges the importance of other non-computer phases of storage and retrieval. Particular aspects of computer applications are reviewed, and a practical example of an information retrieval system, named SARA, for Storage And Retrieval Alberta, and programmed in a new programming language called APL for a time-shared computer,is given. Many aspects of information retrieval are not covered.

Information storage and retrieval systems vary greatly in the degree of mechanization. Semi-mechanized systems utilize machines only in part of the storage and retrieval cycle, usually in the retrieval of information. The preparation of documents for retrieval is done manually in these systems. Fully mechanized systems carry out all processes of the information storage and retrieval cycle automatically. It has not been shown conclusively that automatic indexing is satisfactory in all disciplines, particularly in fields such as history. Figure 1.0.1 pictures in block diagram form a fully mechanized general information storage and retrieval system. Documents containing information must first be converted to a form compatible with the machines. The source of the information

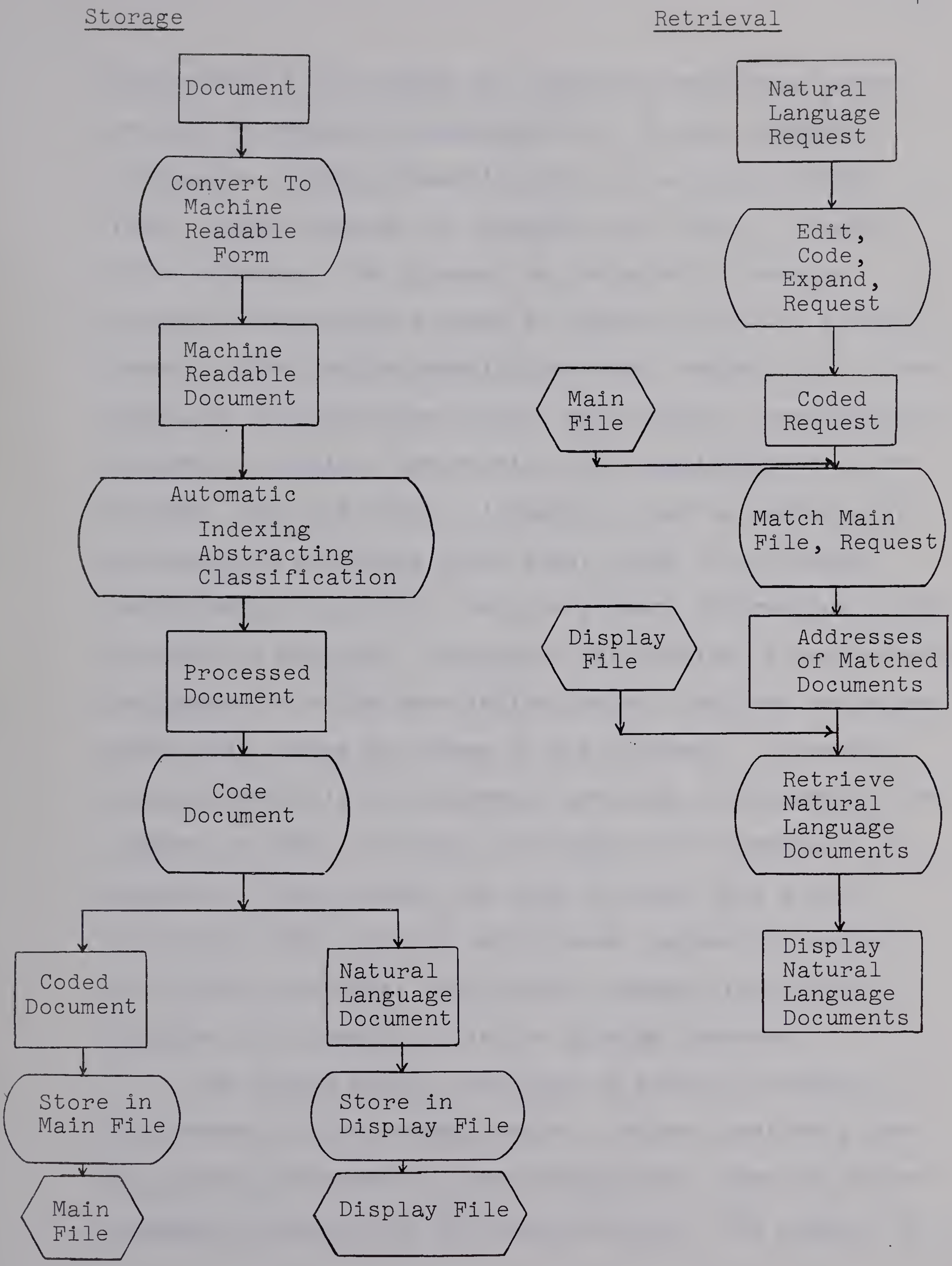
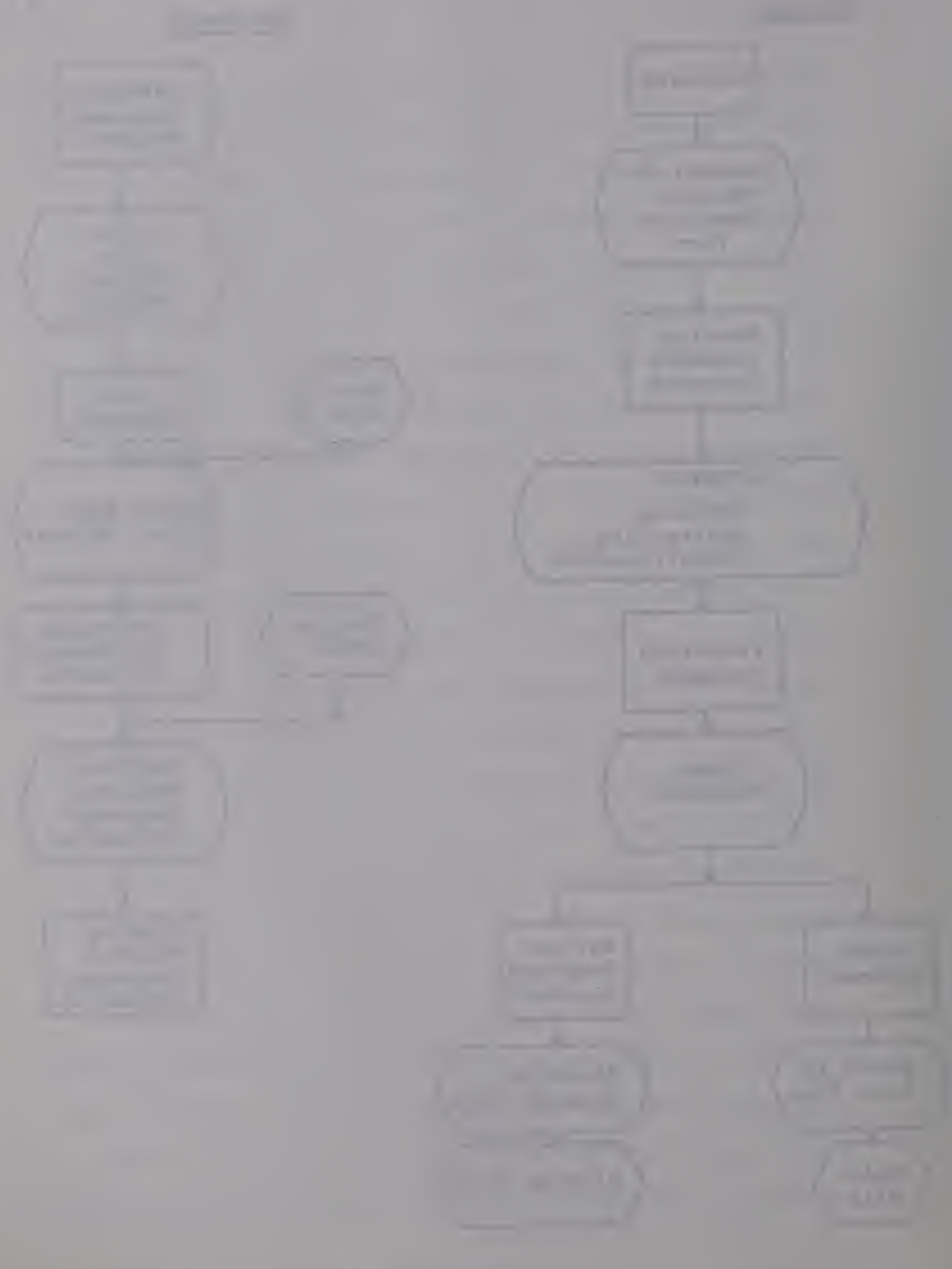

Storage
Document
Convert To Machine Readable Form
Machine Readable Document
Automatic Indexing Abstracting Classification
Processed Document
Code Document
Coded Document
Natural Language Document
Store in Main File
Store in Display File
Main File
Display File
Retrieval
Natural Language Request
Edit, Code, Expand, Request
Main File
Coded Request
Match Main File, Request
Display File
Addresses of Matched Documents
Retrieve Natural Language Documents
Display Natural Language Documents

Figure 1.0.1

Totally Machine Processed Storage and Retrieval



[illegible]

[illegible]

may consist of the words in a book, an article or paper, or even in personal communications. In most instances, the source of the information will be in some printed form. Other examples of documents are voice or fingerprint records. The document is converted to machine readable form, usually paper or magnetic tape, or punched cards. After the information has been changed into a form which can be manipulated within the computer, operations of automatic indexing, abstracting, or classification of the document can take place. Automatic indexing consists of mechanically assigning valid index terms to a document sufficiently accurately that the primary information of the document is retained. Automatic abstracting is the automatic assignment of a few descriptive natural language sentences which best convey the theme of the document. Automatic classification is the automatic grouping of documents into classes in order to reduce the number of documents to be searched. The document can then be coded into a more economical form, such as words being replaced by numbers, for future searching; the natural language form can be retained in a separate file for display purposes.

The second phase, retrieval of stored documents, is the reverse of the storage phase. In most systems a user may request documents by specifying index terms in natural language, connected by Boolean operators. The request is

edited, coded and expanded according to user options, and is matched against some subset of the encoded file of documents. Documents (or addresses of documents) satisfying the match can then be displayed to the user.

The SARA information storage and retrieval system is made up of two parts. The first part consists of the manual selection of documents and index terms and assigning relationships between these terms. The second part consists of a mechanized system which deals with the storage of these documents and their subsequent retrieval. To understand the system, it is necessary to understand some of the difficulties encountered. These are described in Chapters II and III. Chapter II reviews the literature on the conversion of documents to machine readable form. Chapter III describes advances made in automatic indexing, abstracting and classification of documents. Chapter IV reviews seven operational information storage and retrieval systems: three are designed around a batch-processing monitor and four around a time-shared monitor. Chapter V and VI are the most significant. Chapter V deals primarily with the overall plan and the details of the mechanized SARA system. Chapter VI critically analyses the SARA system, suggests possible improvements to it and evaluates the programming language used for the problem. Appendix A describes the manual portion of the SARA system. Appendix B contains block

diagrams and listings of routines used in the mechanized portion of SARA, while Appendix C contains examples of a dialogue between man and machine.

CHAPTER II

TEXT CONVERSION TO MACHINE READABLE FORM

2.0 Introduction

Information can be stored in many ways. The tribal Indians of North America left telltale signs, such as a small cairn of rocks,to inform following comrades about their direction and time of departure from the camp. In medieval Europe, handwritten manuscripts became the principal method of storing information. Presently, the vast majority of information is stored as characters printed on a page. However, information stored in this form cannot be utilized directly by a computer. It must be converted to a form which the computer can manipulate easily, e.g., characters on punched cards or magnetic tape. This chapter introduces two methods of conversion: keypunching and direct source production. The subject of the remainder of the chapter is a review of the research in the field of pattern recognition. Much work has been done on this problem, and much more will be required before a machine will convert a printed page economically into machine readable form. Only the conversion of present forms of storing information into forms compatible with a computer is dealt with in this chapter and no attempt is made to resolve the problem of the meaning of words.

The patterns to be identified in all programs discussed are represented by a square or rectangular matrix of logical elements (Figure 2.0.1). Wherever the pattern coincides with the matrix, the corresponding logical element is activated; otherwise, it remains inactivated. This matrix is examined for characteristics which uniquely identify the input with a name, such as A, B, etc. To the human eye, Figure 2.0.1 represents an A. The problem is to program the computer to uniquely assign the name A to this, and any other figure approximating it. Pattern recognition methods are reviewed under four headings; n - tuple, template matching, random nets and analytic methods.

2.1 n - Tuple Methods

The n - tuple pattern recognition method consists of grouping the input matrix elements into clusters of size n, and then recording the state of each group for a series of pattern names. Thus, if n equals one, each group would consist of two states, activated or inactivated. In general, the number of possible states for a group of n elements is 2^n. A discussion of the case in which n equals one was investigated by Uttley and is described in the reference Uhr (1963). Cases of n equal to two have been investigated extensively, and cases of n greater

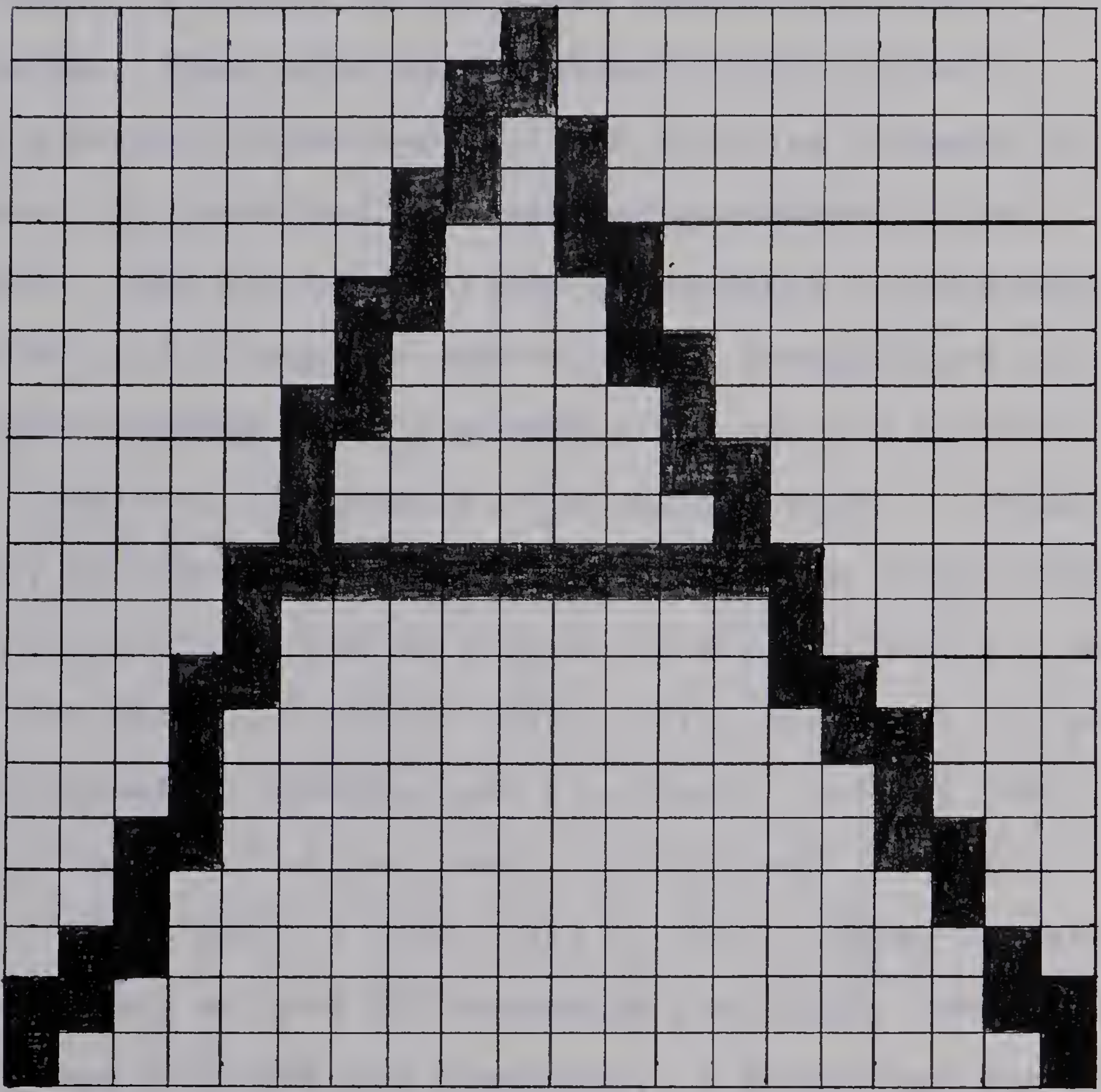
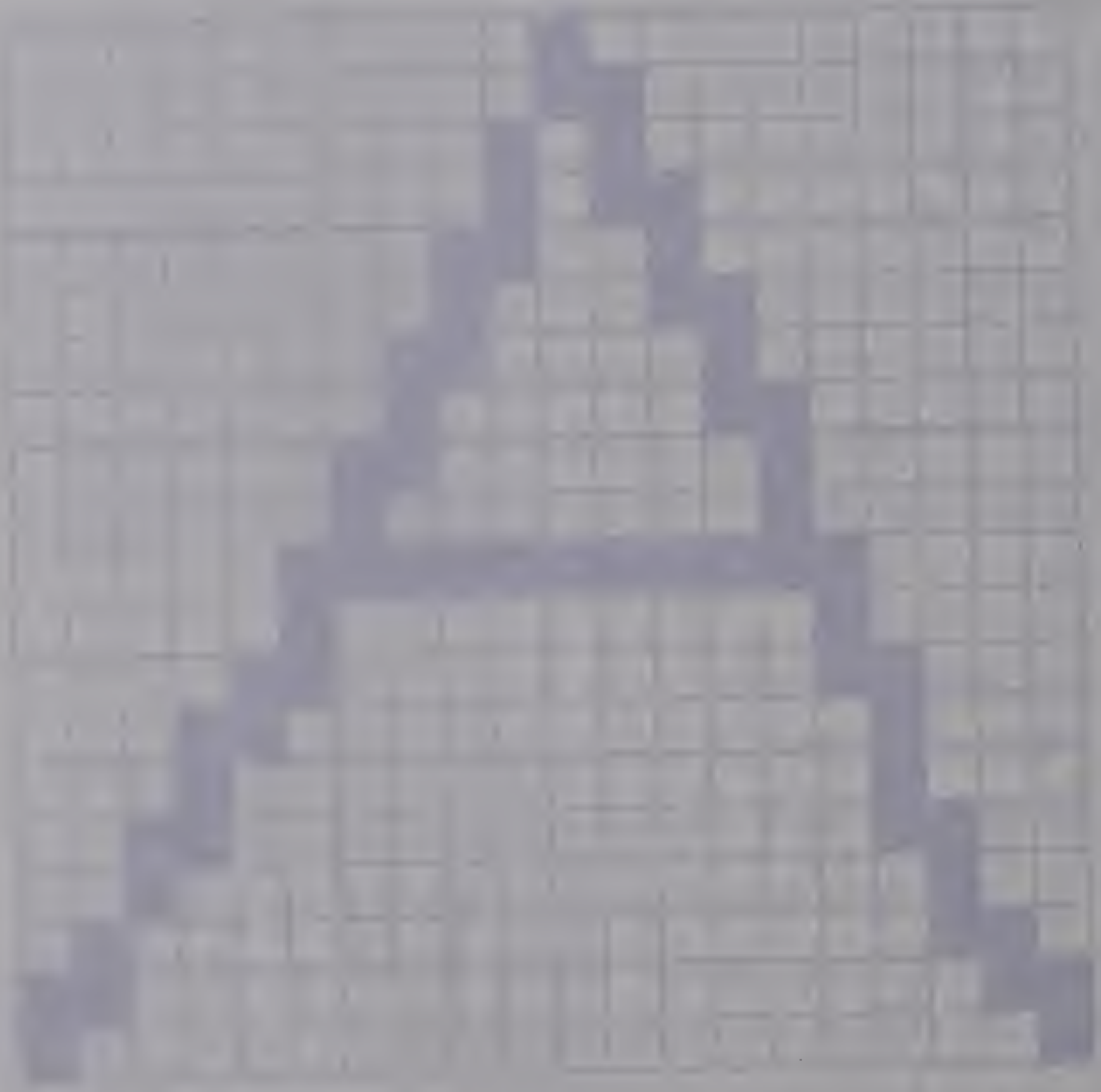


Figure 2.0.1

Matrix Representation of the Character "A"



than two somewhat less extensively (Bledsoe and Browning (1959)).

The n - tuple method examines the relationship between input, the input's name and those combinations of sets of n cells in the input matrix which are activated. These sets may be chosen either randomly or in a predetermined manner. The training proceeds as follows: An input and its name are presented to the machine. Each set of n cells is examined to determine its state. All possible states of all combinations of sets are recorded under the name given for the pattern.

Consider, for example, a situation with n equal to two, as described in Bledsoe and Browning (1959), with an input matrix of ten by fifteen binary photocells. We have, for each of 75 sets of two cells, say cells i and j, four possible combinations of off-on: cell i on, cell j on; cell i on, cell j off; cell i off, cell j on; cell i off, cell j off. Thus, for each pattern name, we have 300 recording positions. When a pattern and its name are presented, 75 recordings are made, i.e., a name for the state of each of the two cell sets.

After a set of patterns and their names has been presented a number of times, an unknown pattern must be identified. This identification proceeds by the presentation

of the input, examination of all 75 sets of two cells, and summation of all possible names over the 75 corresponding off-on states of the input pattern. The name corresponding to the highest score is chosen.

The results of the tests by Bledsoe and Browning (1959) were quite encouraging. The experiment was continued with other modifications, such as normalization of the pattern, examination of the distribution of possible patterns, and consideration of word context. Using hand-written patterns with a large number of names (36), up to 94 percent correct recognition was attained.

2.2 Random Nets

The random net approach is discussed by Brain, Forsen, Nilsson and Rosen (1962) as well as by Rosenblatt (1960). The basic component of the random net approach consists of a unit called the Threshold Logic Unit (TLU). A set of input lines fires into the unit, and, if the weighted sum of these inputs exceeds a specified threshold, the output fires.

One version of the random net approach (Brain, et al. (1962)) is that of the Perceptron (Figure 2.2.1). Three sets of Threshold Logic Units, referred to as S, A, and R units, are used. The first set of 35 sensing units (S units) is constructed as an input matrix. Each of ten

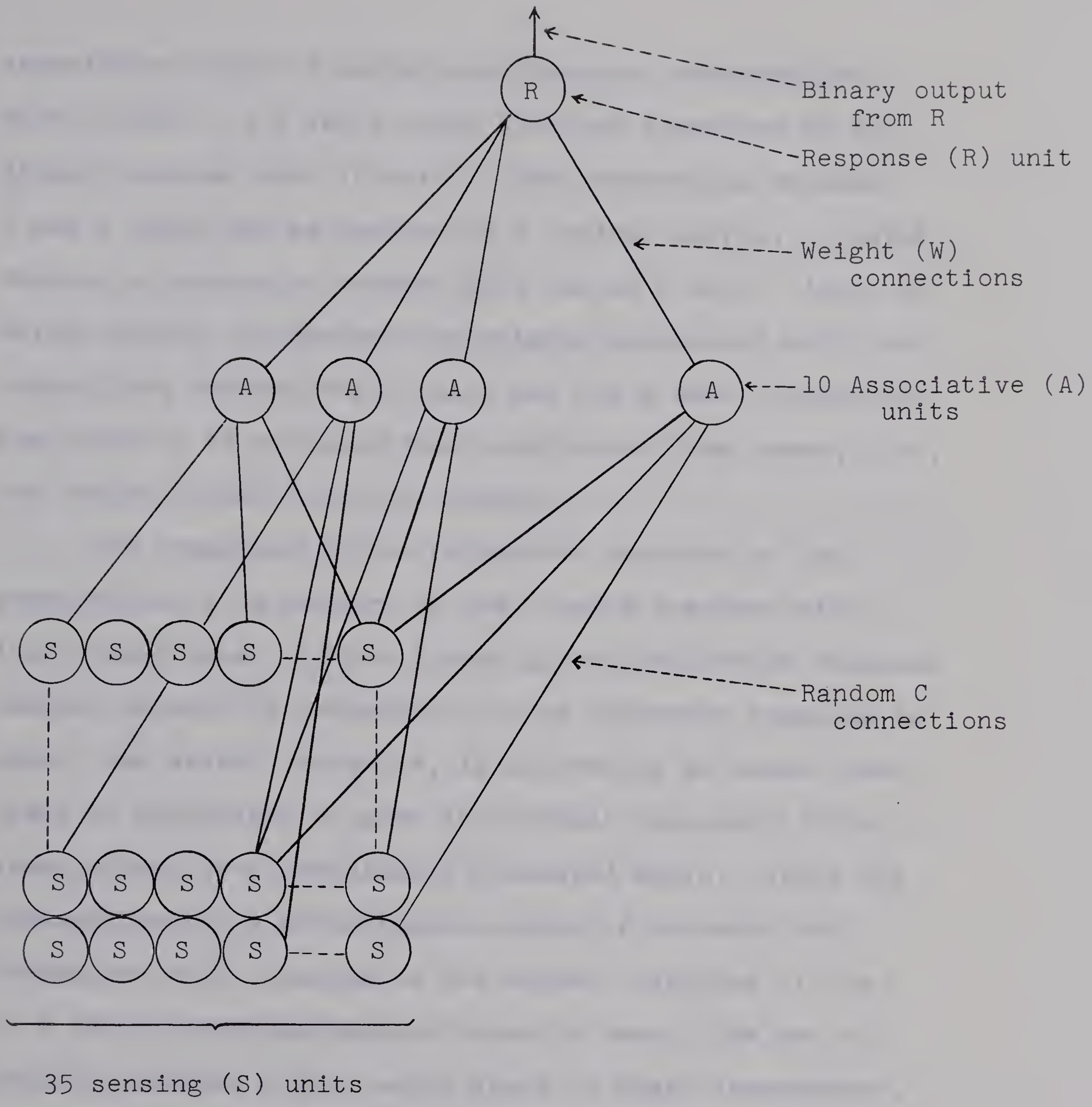


Figure 2.2.1

Schema of a Perceptron

associative units (A units) are randomly connected to nine S units. All ten A units are then connected to a single response unit (R unit). The connections between S and A units can be denoted by a logical matrix, C, which denotes a connection between an A and an S unit. Also, a weight vector, W, denotes the weights associated with the connections between the A units and the R unit. Note that the input is to be classified under one of two names, i.e., the output of the R unit is binary.

The training of this Perceptron consists of the presentation of a pattern to the S units together with its correct name. If the R unit gives the correct response another pattern is presented. If an incorrect response is given, the weight vector, W, is altered by an amount that would be sufficient to give the correct response if the same pattern were immediately presented again. After the presentation of a determinable number of patterns the Perceptron will converge to the correct response if one of a set of error-correction rules is used. The set of error-correction rules, and a proof of their convergence, is given by Nilsson (1965). Brain, et al. (1962) cite another example of a random net approach called Madaline. This random net approach, developed by Widrow, consists of the same arrangement as found in the Perceptron. However, instead of the W vector being altered in the training

procedure, the C matrix denoting the connections between A and S units is altered. The W vector remains constant. No concrete results of tests are given in this paper; it is felt that better results can be obtained by less random techniques.

Roberts (1960) did some experimental work with random nets, which duplicated and extended the work done on the Perceptron. By using a modified error-correction rule, and altering the W vector, Roberts achieved up to 94 percent correct classification on a set of 44 characters. A further constraint was imposed by Roberts in order to achieve this high degree of correct response. Instead of completely random connections, the C matrix was preset by the author such that all S units were uniformly distributed over the A units. This choice of the C matrix, along with the error-correction rule used, produced the effect of recognizing spatial nearness in the device, an effect which the Perceptron and Madaline failed to achieve.

The random net approach emphasizes parallel, as opposed to sequential, processing of information. In parallel processing all information is gathered at each stage; a decision is not made until all possibilities have been calculated. The next stage of the classification is then initiated. In sequential processing a calculation is made and a decision is arrived at after

each calculation. The choice of a superior method depends on the cost of making a decision versus the cost of making a calculation.

Selfridge (1959) stresses parallel processing. A set of cognitive and computational "demons" are connected in layers, each layer being equivalent to the A units of the Perceptron. The initial layer, referred to as data demons, corresponds to the S units, and the decision demon corresponds to the R unit. All demons of each layer are connected to all demons of the next layer. There is no randomness in these connections. Various training procedures are described, but no results are given.

2.3 Template Matching

Template matching is the easiest and most widely used commercial method for pattern recognition to date. However, it is quite limited in its scope. The method as outlined by Minsky (1961) consists of two phases, the first being the normalization of the input pattern. This is achieved by changing the relative size of the pattern to match the size of internally stored pattern, and the rotation of the pattern about some point, usually its center of gravity, in order to orient it to the internally stored replica. The data may also be smoothed. The second phase consists of matching the normalized pattern against

a previously stored set of all the possible patterns. Similarities are calculated for all patterns, and the name with the highest similarity score is chosen.

There are many drawbacks to such a system. A prototype of all possible patterns must be available to the machine prior to identification. A set of similarity tests must be programmed into the model. Abstract classes of patterns, such as all patterns with three intersections of straight lines, cannot always be handled. Slight variations of the input patterns are critical to correct identification.

However, it will be noted that, to date, this is the principal method employed by commercial machines. Subject to the above limitations, the percentage of correct identifications is extremely high. If the percentage of correct identifications is the only criteria of success then this method rates high among all the methods reviewed here.

2.4 Analytic Methods

The analytic methods developed to date most closely parallel the observed functioning of human processing of information. Gyr, Brown, Willey and Zivan (1966) suggests a recognition algorithm which only recognizes straight lines. The algorithm is original in that it attempts to

simulate the observed behavior of humans as closely as possible without regard to the efficiency of the algorithm. The input, a 144 by 144 logical matrix, is scanned by a smaller matrix called the retina. This is a 36 by 36 operator matrix which is divided into a periphery (the outer section of the retina), and the fovea (the main detector). As the retina moves across the pattern, it is directed along the straight line by the fovea. The scan is divided into two parts. If a "quick look" criteria is satisfied, then the scan continues; if not, a "close look" is initiated, and a decision is made on whether it is still on a straight line. Small amounts of noise can be tolerated.

Other more elaborate systems have been programmed to recognize more than sections of a pattern. Selfridge and Neisser (1960) developed a program which, after clean-up and normalization of a pattern, inspects features of the pattern, and ranks each pattern by its similarity with respect to these features. During training, 28 features such as "the maximum intersection with horizontal line", "concavity facing south", and "length of the south edge" are inspected for each of the ten possible patterns, and probabilities of occurrence are calculated for each pattern for each feature. Upon presentation of a pattern to be identified, all such features are inspected, and the

probabilities for all patterns are summed up. The name corresponding to the largest sum is assigned to the pattern. No results are given. In the programs discussed above, all tests performed are programmed into the model. In the experiment discussed below not only are the characterizing operators evaluated, but they are generated by the program itself.

Uhr and Vossler (1963) developed and tested a program which would generate and evaluate operators, and then discard the useless ones. The input consists of a 20 by 20 binary matrix which is scanned by a five by five operator matrix. This operator matrix is generated either randomly or deterministically, and characteristic strings of the patterns are generated. These strings serve to retain, as well as to generalize, the learned patterns. Records of success for the various operators are kept, and those of little use are discarded. Amplifiers, which are used in general as well as in local discrimination functions, are adjusted, and serve to discriminate between the patterns. The program was tested on hand printed and written characters as well as voice patterns. There was a high degree of success after ten training samples. A revision of this work (Praether and Uhr (1964)) appears to be less sensitive to noise and to the thickness of the pattern, although the results of the tests are obscure.

Grimsdale, Sumner, Tunis and Kilburn (1959) approached the problem in a different way. Each pattern was first divided into various components by a scan, and then analyzed as the components were reassembled. More information was retained about the topology of the figure than by previous methods, and the system was relatively insensitive to orientation of the pattern. The approach involved an analysis of the pattern as a whole since information about the form of each part and its connection with the other parts of the pattern was retained.

2.5 Discussion

There are many approaches to solving the problem of creating machine readable text. The most general method, that of pattern recognition, has advanced rapidly since it was first proposed. However, any commercial system now available is not only expensive but cannot recognize the wide range of type fonts which would be required of it. The most promising technique at this time appears to be capture of the data at the point of publishing. Some publications, such as Chemical Abstracts, presently provide such a service. Magnetic tape copies of the abstracts provided by Chemical Abstracts are available. Cooperation among publishers and users in this direction may yield the most benefit in solving the problem of producing machine readable text.

The field of history is of particular concern in the present investigation. Since there is already a great deal of information in printed form, a solution to the problem of character recognition would be of great value.

CHAPTER III

AUTOMATIC ANALYSIS OF TEXT MATERIAL

3.0 Introduction

One of the most important phases in the information storage and retrieval cycle is that of preparing documents for subsequent retrieval. Chapter II indicated one manner of preparing documents. Another way could consist of assigning index terms, or descriptors, to documents. An index term describes part or all of the content of a document. For example, the index terms "church", "economics", "politics" and "1930" may describe a document entitled "The Political Impact of Church Estates in 1930". Retrieval of such documents depends on the appropriateness of the index terms assigned to the documents. The document must be described as concisely as possible for efficient retrieval, yet at the same time as comprehensively as possible for retrieval in the future. Hence, the text must be described in terms sufficient for both present and anticipated needs.

The digital computer has long been recognized as a device which could be used to analyse text automatically. With the computer's very powerful arithmetical and logical capabilities, a statistical analysis of machine readable text material becomes possible, once the type of analysis has been determined. With the large memories of the machines

of today and with the economic feasibility of ever increasing memory sizes, machines can perform logical operations among large numbers of words of text. Moreover, complex table lookups, which can be useful in automatic analysis of text material, add to the power of a computer.

The measurement of the effectiveness of such indexing is in itself a large problem. Although comparison with human indexing is the most obvious method of measurement, it is not entirely satisfactory: humans cannot always agree on how a document should be indexed, abstracted, or classified. Criteria unrelated to human measurement should be set up in such a way that the aims of automatic indexing are satisfied, viz, such that the indexing terms used to describe the article result in high relevance to the documents retrieved, or in comprehensive classification. Ultimately, the effectiveness of the indexing will be determined by the appropriateness of the retrieved documents.

Presently, much manual effort is being expended on the indexing, abstracting, and classification of documents. As the volume of material to be stored increases, more trained personnel will be required and hence, investigation into the automation of these tasks can be justified.

3.1 Automatic Indexing

In all phases of automatic analysis of text, some form of automatic indexing is used. Most of the methods used to

perform automatic abstracting and classification eventually depend on a choice of index terms which describe the text. If the methods are to be effective, the choice of index terms is critical.

The methods to be described for choosing informative index terms are dependent on word frequency. Some methods also use auxilary information, such as the frequency of occurrence of word groups and the function of the word in the sentence.

Pioneering work in the field of automatic indexing was done by Luhn (1958). Using an IBM 704 computer he analyzed scientific and technical text punched on cards, and produced a list of significant words ranked by frequency of occurrence in the text. Luhn reasoned that the frequency of occurrence of a word root in the text was a measure of its "information power". For example, "differ", "differentiate", "difference" and "differently" are all of the same root. In calculating the information power of these words, all forms of "differ" would be considered identical.

A frequency count of all the words resulted in a curve similar to that given in Figure 3.1.1. The words of high frequency such as "the", "a", etc., constitute noise, and could be eliminated by a table look-up procedure. Walston (1965), in reviewing the work of Luhn, suggested that a high frequency cutoff through statistical analysis

could also be used. This is line C in Figure 3.1.1. The remaining words were then ranked by frequency, and those of highest frequency were chosen as index terms (between lines C and D in Figure 3.1.1). Luhn further reasoned that the information power of the words bracketed by lines C and D was represented by curve E. Thus, the list of words produced would constitute a representative picture of the text analyzed. This method is oriented toward, and easily implemented upon, computers. However, any information contained in the grammar or syntax of the article, or groupings of words, is ignored.

Baxendale (1958) compared three methods of automatic indexing. The primary aim was to decrease the amount of text analyzed, and still retain as much as possible of the information contained in the entire text. A set of six papers from six different scientific journals was used to test each of the three methods. In all three methods, the technique of deletion of common words before analysis of the text was used to decrease the noise factor. The first method, similar to Luhn's, was an analysis of the entire text, with subsequent ranking by frequency of the remaining words after deletion of common words. The second method was an analysis of the topic sentences of each paragraph. A previous analysis indicated that 85 percent of the topic sentences occurred as the first sentence, while seven

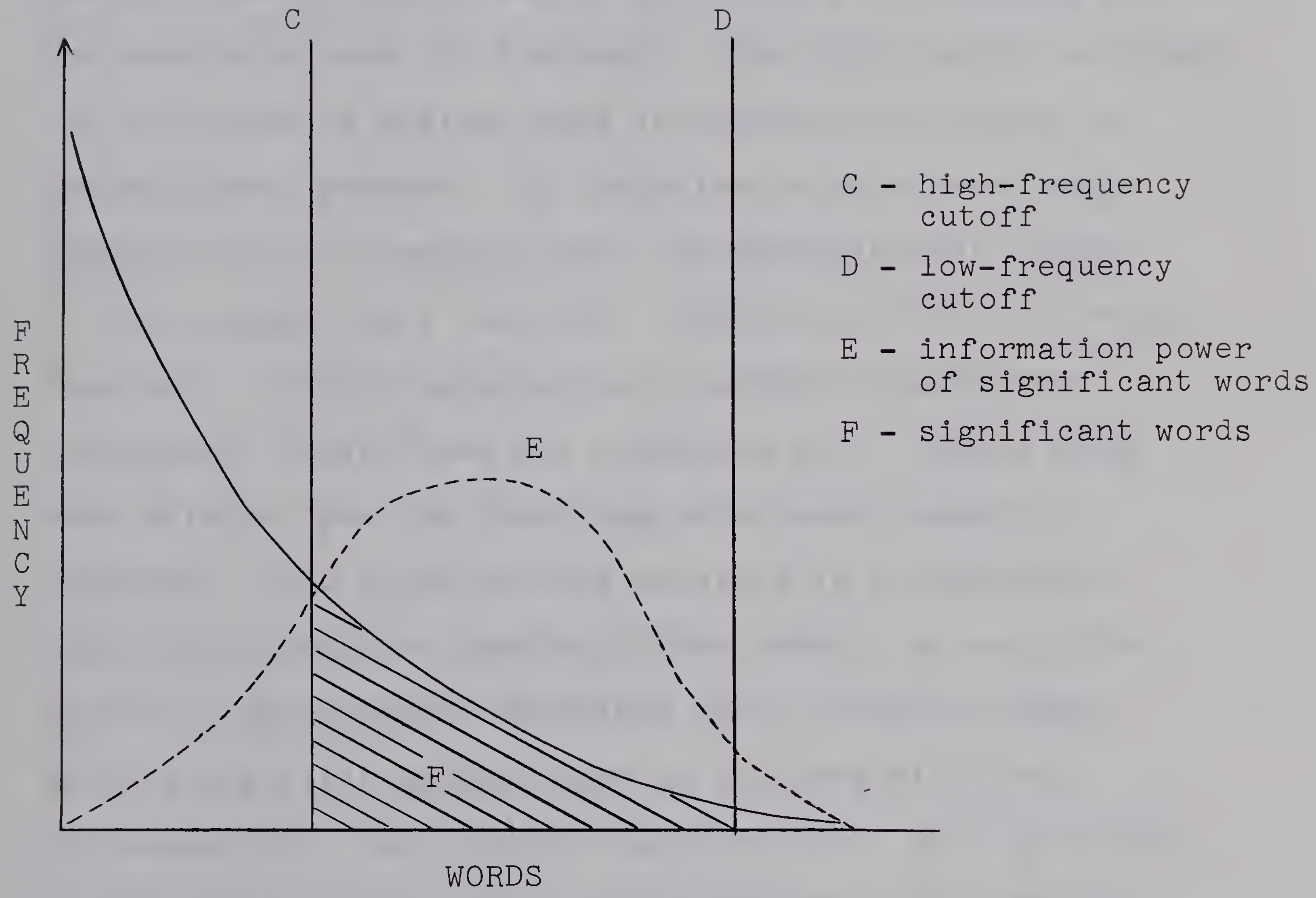


Figure 3.1.1

A Word Frequency Diagram

percent occurred as the last sentence, of each paragraph. The second method consisted firstly of the selection of the first and last sentence of each paragraph, secondly, the deletion of common words, and thirdly the ranking of the remaining words by frequency. The third method utilized the fact that in English much information is carried in prepositional phrases. By comparison with a previously compiled list of prepositions, the prepositional phrases of the document were isolated. Selection of the following four words (unless punctuation or another preposition intervened) constituted the selection set. Common words were deleted, and the remaining words were ranked by frequency. The three methods resulted in a remarkable similarity among the ranking of the words. By using the second or third method described above, results comparable to the first method could be attained with less processing and less machine readable text. As a by-product of the third method, terms used together in the original article remain coordinated to a certain degree, thus retaining some of the syntax of the original article.

Edmundson and Wyllys (1961) used a different approach to the problem of selecting index terms to describe a document. They advanced the argument that the information contained in a word is inversely proportional to the frequency of occurrence of the word; thus, the rare or

unusual words in an article give the greatest indication of its content. However, the word must be rare in general usage, not rare within the article itself. Four significance factors for each word are suggested:

$$s_1 = f - r \tag{3.1}$$

$$s_2 = f / r \tag{3.2}$$

$$s_3 = f / (f + r) \tag{3.3}$$

$$s_4 = \log (f / r) \tag{3.4}$$

where s = the significance factor of a word;

f = the relative frequency of a word within the document;

r = the relative frequency of a word in general use.

These significance factors are analyzed to determine which have the greatest relevance to the document being indexed. The author concludes that the functions $s_1 = f - r$ or $s_2 = f / r$ are the most relevant, though no experimentation was carried out by the author. This method does not require that common words be deleted. The significance function will handle these.

A further classification of the article into various spatial categories, such as title or introductory paragraph,

may be used to add weights to the significance functions. These weights reflect the amount of information that the particular word carries by virtue of its position in the article. The final significance function may then be calculated as

$$(3.5) \qquad s_f = b_1 b_2 b_3 s(f,r)$$

where $b_1, b_2, b_3 \geq 1$ and

$$b_1 = \begin{cases} b_t & \text{if the given word occurs in the title} \\ 1 & \text{otherwise} \end{cases}$$

$$b_2 = \begin{cases} b_f & \text{if the given word occurs in the first paragraph} \\ 1 & \text{otherwise} \end{cases}$$

$$b_3 = \begin{cases} b_s & \text{if the given word occurs in the summary} \\ 1 & \text{otherwise} \end{cases}$$

The terms b_t, b_f, and b_s are arbitrarily assigned weights determined by experience.

An experiment was conducted by Damereau (1965) on the criteria suggested by Edmundson and Wyllys. Eight articles on world politics appearing in Atlas magazine were indexed for testing purposes. The frequency of occurrence of words in general use were obtained from approximately one million words of radio news broadcasts in the field of world politics.

He first indexed a series of articles manually, and then indexed them using a Poisson probability function. The index terms for the article were chosen as those words which occurred sufficiently often that the probability of such a frequency of occurrence, in general usage, is less than or equal to 0.0005. In addition, the three functions, $s_1 = f - r$, $s_2 = f / r$ and $s_3 = f / (f + r)$ were calculated, and the list of words chosen by each function were compared to those chosen manually as well as by the Poisson distribution function. The results indicated that the Poisson criteria minimized both the number of extra words chosen and the number of index terms missed. Damereau also pointed out that the approach taken by Edmundson and Wyllys is very difficult to test and implement because a universe of terms must be created with which to compare the words of the documents. Similarly, before the efficiency of the SARA system can be judged, a suitable universe of terms must be compiled which will suit the needs of the application, i.e., history.

A common and valid criticism, such as that given by Bourne (1963), of the techniques described above is that the frequency of occurrence of words is the only criteria used in choosing index terms. No consideration (except, perhaps, the prepositional phrase method of Baxendale) is given to the syntax, order, or grouping of the words.

There are difficulties, however, in determining how significance should be attached to such information. Word groupings, as well as their frequency, have been investigated by Oswald (see Edmundson and Wyllys (1961)). His treatment is an extension of the work of Luhn and Baxendale.

Tests must be made to measure the significance of the index terms. Some criteria, such as comparison with a human indexer's results, should be satisfied. This has been done in the above experiments by using manually generated indexing terms in automatic abstracting. A study of the results given by the SARA system and the reaction of users to these results could indicate the appropriateness of the index terms chosen for the system and improvement in the choice of terms.

Another concept that should be considered is that of attaching weights to the indexing terms. These weight factors may be determined by the frequency of occurrence of the corresponding index term, and signify the relevance of the indexing term to the article. The position in the hierarchial classification system that the word occupies could also be taken into consideration; the nearer the term is to the root of the hierarchial tree, the more general it is, and hence the less significant.

3.2 Automatic Abstracting

The next step in the automatic analysis of text is that generally referred to as automatic abstracting (or, more accurately, automatic extraction of representative sentences). In automatic indexing, a significance function is calculated for a word; in automatic abstracting a significance function is calculated for a sentence. The sentences are then ranked by their significance factors and the high ranking sentences are used to form an abstract of the article. The significance factor attached to each sentence is a function of the selected index terms. It is a logical extension of calculation of the significance factors for individual words.

Luhn (1958) calculated significance factors of sentences by the following method. Each sentence was scanned to determine if it was bracketed by previously obtained significant words. If no more than five non-significant words intervened between significant words, the grouping was considered significant. The significance factor was then determined by squaring the total number of significant words in the resulting cluster, and dividing by the total number of words in the cluster. The highest ranking cluster of a group of clusters in a sentence was used as the significance factor of the sentence.

A limitation to this technique becomes evident if a sentence has several low ranking clusters. Such a sentence

will not be chosen over a high ranking single clustered sentence although more information may be contained in the former. Edmundson and Wyllys (1961) suggest a combination of both word grouping and number of clusters within a sentence as the criteria for selection (modified by sentence length). They suggest a function, E, of the significance factor s, such that $E(s) > 1$ for large s, $E(s) \approx 1$ for medium s, and $E(s) < 1$ for small s. The significance factor could then be a combination of the occurrence of two significant words in the text, their position and the number of occurrences within a document. The results of both tests are quite encouraging. Although some of the abstracts lack continuity, the idea of the article is conveyed.

3.3 Automatic Classification

Automatic classification takes automatic indexing one step further, and organizes a mass of unrelated data (index terms) into an hierarchial structure which aids the computer in its subsequent retrieval request. The problem is one of entering the document under the proper index terms so that a search request will subsequently retrieve the relevant document. Again, automatic classification is determined by the index terms selected by the indexing procedure.

There are two prevailing methods by which to approach the problem of assigning documents to a classification system. One is that a classification system first be drawn up, and that documents then be assigned to these classifications (an _a priori_ system). The other is that a system of classification be derived from a set of test documents and that subsequent documents be classified with respect to these classes.

A third method, one which will become more important in the future, was suggested by Becker and Hayes (1963). The classifications could be automatically reorganized from time to time, depending on the current uses of the file. This method implies that statistics be kept on the use of the system, and that the system be reorganized periodically.

Maron (1961) pioneered some work along the _a priori_ lines. His approach can be divided into two parts. The first part consisted of selecting a set of documents - in this case, 405 abstracts from the _IRE Transactions on Electrical Computers_, Volumes 2, 3 and 4, (1959) dealing with the field of computing. These were divided into two groups: Group One was used to determine the categories and index terms while Group Two was used to validate the results of Group One. Thirty-two categories best describing these documents were drawn up manually (the classification scheme), and the documents of Group One were indexed according

to these categories. A group of clue words was chosen which best described the documents. A category versus clue words matrix was set up by sorting the documents into their proper categories. A correlation matrix was calculated which gave the correlation of each clue word to every other clue word. In the second part, the documents of Group Two were used to test the effectiveness of this set of clue words and the categories. These validation documents were automatically indexed by frequency of occurrence of index terms, and their index terms were used to place the document into the proper classification. The author used a Baysian approach to predict to which category a document would belong by the clue words it contained. The Baysian prediction formula was used in the following form:

$$(3.6) \quad P(C_j|W_1,W_2,\ldots,W_n) = \frac{P(C_j)P(W_1|C_j)\ldots P(W_n|C_j)}{P(W_1)P(W_2)\ldots P(W_n)}$$

where $P(C_j|W_1,W_2,\ldots,W_n)$ = probability that a document belongs to class C_j given the occurrence of clue words $W_1,\ldots,W_n$;

$P(C_j)$ = probability of class C_j;

$P(W_i|C_j)$ = probability of word i occurring in class C_j;

$P(W_i)$ = probability of word i occurring at all.

The formula (3.6) is valid subject to the assumption that the clue words occur in a statistically independent manner.

Maron's experiment resulted in 84.6 percent of the documents of Group One being classified the same as an independent classification by humans, whereas the Group Two classification corresponded to 51.8 percent of the manual classification. These results demonstrated that the classifications were better than chance, though not yet as accurate as manual classification.

Borko and Bernick (1963 and 1964) followed up Maron's work by conducting more experiments on the same set of 405 abstracts and comparing results. Categories were generated by applying factor analysis to the correlation matrix of index terms (not an _a priori_ system). They contended that a mathematically derived classification system would be more descriptive of the class of documents and more amenable to automation than an _a priori_ system. This system would also be independent of time, i.e., as more and different index terms occur and as new categories arise, they may be incorporated into the system without difficulty. In the experiment described below, a manual classification of the documents was used as a control group.

Three hypotheses were tested as follows. First, by using the Baysian prediction formula on the _a priori_ categories, a more accurate classification will result than if

a modified factor score resulting from factor analysis is used. Factor analysis was performed on the 90 index terms suggested by Maron. Twenty-one categories were derived, and these categories were used to classify both Groups One and Two. These results indicated that the two automatic classification systems agreed to a large extent on both Group One and Group Two, but that the results were not as impressive when applied to Group Two alone. However, there was enough evidence that automatic classification is better than random classification.

To test the second hypothesis, Maron's list of descriptors was replaced by one compiled by Borko. This new list of words in Group One descriptors was derived by a frequency count of the significant words in Group One. A correlation matrix denoting the correlation between index terms was factor analyzed into 21 different categories. Then, the test used with Hypothesis One was applied to these new index terms and categories. The results relating to Group One revealed an increase in correct classification (by manual standards) by both Baysian and factor score methods. On Group Two, the Baysian method correctly classified 55.9 percent. Hence, both methods are approximately equally effective in classifying documents.

The third hypothesis tested was that a classification set derived from factor analysis would result in a larger

percentage of correct classification than an a priori system. The results verified that this was true to a greater extent when applied to Group Two documents (the important set) than to Group One documents. The results of the experiments by Borko and Bernick indicate that automatic classification of certain types of documents is feasible. The limiting factor is the quality of the indexing.

Doyle (1965) published a survey of present classification techniques which differentiated between two trends of thought, viz, automatic indexing, which implies a complete search of all stored documents to satisfy a retrieval request, and automatic classification, which aids in limiting the search and in organizing the data. These two points of view were reconciled and this discussion was followed by an analysis of a method to increase the quality of classification. An experiment was described which indicated that higher quality classification results as more information about each document is retained. From 12 to 36 indexing terms were used to describe each document. The test consisted of measuring the indexing capabilities of each set of indexing terms against six human criteria by varying the number of index terms retained, beginning at 12 and increasing to 36. Four of the criteria were satisfied as more information was kept about each document. To increase the quality of the classification, the amount

of information retained should be increased. This increase will also increase the storage requirements accordingly.

Automatic classification has distinct possibilities. If classification could be mechanized so that 75 percent accuracy were attained (as compared to the manual system) then machine capabilities would approach those of humans. Presently, there is insufficient data available to automatically classify documents in the field of history.

3.4 Other Areas of Computer Analysis of Text

Once information is in machine readable form, computers can be used to produce a concordance of text material. A concordance is an alphabetic analysis of the text, word by word, together with details of the location in which each word is used. Some of the first work in the area of computer produced concordances was by Tasman (1957). A complete concordance of *Summa Theologiae* by St. Thomas Aquinas was compiled from 1.6 million punched cards. The occurrence of each word was summed, and a printout gave the frequency of occurrence, word usage, and place of occurrence of each word. Later, Silva and Bellamy (1965) experimented with a concordance generator. In it, English and French text could be processed, and a selected concordance of the text could be produced. Word counts and an index were also produced.

Linguists are finding many uses of, and variations on, the computer produced concordance. It may well be that the next step will be completely generalized machine independent concordance generators free of the upper case typing limitation of present day computers.

3.5 Discussion

Once text material is in machine readable form, the application of a computer to index and classify documents automatically seems not only feasible, but highly desirable. Although computer indexing may not be as accurate as manual indexing, computers have the advantage of consistency, and, to a large extent, of predictability. Much more must be done in this area if indexing and classification are to keep abreast of the present information explosion that is occurring in both scientific and non-scientific literature.

There are many reliable papers and books on the subject of automatic analysis of text. Walston (1965) gives a wide survey of the methods used to date; in text form, Becker and Hayes (1963) present the librarians' point of view. Bourne (1963) has a very readable survey of present methods together with an excellent bibliography. Vickery (1965) presents the material well in a more sophisticated and formal way than Bourne.

CHAPTER IV

OPERATIONAL INFORMATION STORAGE AND RETRIEVAL SYSTEMS

4.0 Introduction

Many information storage and retrieval systems are presently implemented on computers. A selection of these, representative of the current state of information systems, will be reviewed in this chapter.

The systems fall into two categories: those implemented in a batch-processing environment, and those designed for a real-time time-shared environment. In most applications of the first category, two functions are to be performed: to aid in the publication of abstract and announcement journals, and to do retrospective searching. The real-time systems are used primarily to satisfy a request for information. Other ancillary functions, such as the Selective Dissemination of Information, can be implemented on these systems.

The batch-processing systems to be reviewed are the PICUPS system at the U.S. National Agricultural Library, the MEDLARS system at the U.S. National Library of Medicine, and the HAYSTAQ system at the U.S. Patent Office. The real-time systems are the CONVERSE system at Lockheed Missile and Space Company, the TIP project at Massachusetts

Institute of Technology, the SMART system at Harvard University, and the BOLD system at System Development Corporation.

4.1 Batch-Processing Systems

The three systems to be reviewed are typical of the implementation and aims of batch-processing information systems. They function as a mechanical aid to production of publications as well as to retrospective searching. However, the HAYSTAQ system is used exclusively for retrospective searching.

It is not yet economical to dedicate an expensive computer system exclusively to the searching of literature. Hence, most current systems are justified economically on their added benefits, such as production of publications. Input to the batch-processing systems is through a card or paper tape reader, and output is to magnetic tape (for offset printing or typesetting) or to a printer.

4.1.1 The PICUPS System

The PICUPS (Pesticides Information Center; Update, Publication, Search) system developed for the National Agricultural Library by Datatrol Corporation (1965) is designed with two purposes in mind: retrospective searches to satisfy inquiries, and the publication of announcement journals. It is designed with a maximum of flexibility to

allow changes and modifications to the system with a minimum of disruption. The PICUPS system required an estimated 470 man-days of programming time. The time spent from initial feasibility investigation to completion totaled 15 months.

The documents to be stored in the PICUPS system are indexed to two levels by a professional staff: the first level is for publication and the second is for retrospective searching.

The indexed documents are typed on a form and manually edited, and the corrected forms are converted to magnetic tape by an optical scanner. The vocabulary in the PICUPS system is well controlled, with generic structuring and many cross references. New terms are added as the need arises. The vocabulary file can be updated easily with additions and deletions of terms and cross references.

There are two files of bibliographic references and associated descriptors. The Issue File contains all the recently indexed issues of journals and articles. It is less voluminous than the Master File, which contains all the bibliographic data ever processed by the system. Periodically, the Issue File is used to update the Master File.

Information for the publication of announcement journals is retrieved from the Issue File. The file is scanned, and a magnetic tape is produced containing all

the information necessary for the publication of the announcement journals. The magnetic tape is in a format suitable for input to the Linotron, a typesetting machine. Hence, all publications are of high quality print, with a wide variation in type fonts.

Retrospective search queries use weighted index terms connected by Boolean operators. Requests are screened by one out of eight professional staff members; they are then mechanically expanded to include generically lower terms and cross reference terms. The Master File is searched for document titles that satisfy the requests and they are output on magnetic tape for offline printing on the computer's printer.

The PICUPS system is a standard information storage and retrieval system; it uses magnetic tapes as storage, and scans the entire file in response to a request. User feedback to the system is almost nonexistent, the vocabulary is well controlled, and the search strategies are straightforward and uncomplicated.

4.1.2 The MEDLARS System

The MEDLARS information storage and retrieval system (General Electric Company (1963)) has two functions. It is used for retroactive bibliographic searches on stored documents and it produces many of the publications of the

National Library of Medicine, Washington, D.C. Index Medicus, Cumulated Index Medicus and the List of Journals Indexed are but a few of these publications. Preliminary investigation and design of the system required three months; implementation required an additional two years.

The data base of the MEDLARS system consists of journal articles and published monographs, both English and foreign, which are well indexed by a staff of professional indexers. This indexed information is then transferred to paper tape and a typewritten sheet by a Friden Flexowriter, and the data are edited manually. In particular, consistency and thoroughness in indexing is checked. The paper tape is then edited by the computer, and unit records, one for each article or monograph indexed, are output on magnetic tape. This tape file is subsequently used for updating the Master File of unit records, and also for publication of Index Medicus. A set of magnetic tape files exists which contains all information necessary to verify indexing terms as well as journal titles, etc., used for the publications.

The edited unit record file is sorted by subject heading, and reformatted to a form compatible with an off-line printing device called GRACE (GRaphic Arts Composing Equipment). GRACE is a system which produces justified photographic copies of pages suitable for offset printing.

The publications produced by GRACE, such as Index Medicus, are of high quality print, many type fonts, and full justification.

The entire Master File of unit records may be used in a retrospective search. A search request, submitted as an expository English paragraph, is coded by an information specialist familiar with the MEDLARS system, its operation, and the index terms in a form compatible with the computer. This request is then batched with other requests, and the entire unit record file is searched. However, instead of attempting to satisfy every search in its entirety on the first pass, a second file of unit records, much less voluminous and more easily manipulated than the entire Master File, is produced by the initial scan. This screened file is then rescanned, and all documents satisfying the requests are output on a magnetic tape suitable for either off-line printing on the computer's printer or for offset printing on GRACE.

The search request is formulated as a set of coded index terms connected by Boolean operators. For example,

$$R: (M1 + M2) * (M3 + M4)$$

can be used to retrieve documents containing coded index terms (M1 and M2) or (M3 and M4); + corresponds to "and" and * corresponds to "or".

Like most current commercial information systems the data base is not restricted to retrospective searches. The publication of an abstracts journal of recent literature such as Index Medicus is often the prime objective of such a system, rather than the retrospective searches. MEDLARS is designed to increase the quality of Index Medicus, as well as speed its publication. Retrospective searches are a secondary result of having machine readable documents. Also resulting from the mechanized system is a large body of medical information stored in a machine readable form and suitable for analysis by organizations other than the National Library of Medicine. An article in the Journal of Data Management (1966) gives an example of the use of unit records received from the MEDLARS system in schizophrenia literature.

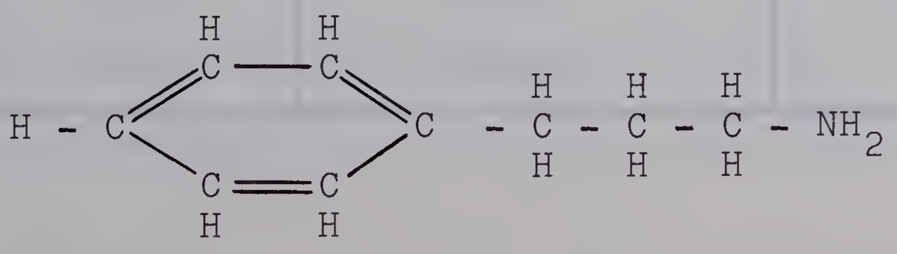
4.1.3 The HAYSTAQ System

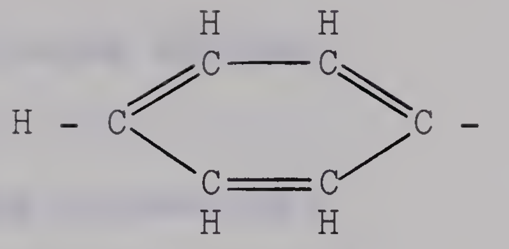
The HAYSTAQ system, developed by the U.S. Patent Office (Marden (1965)), is used exclusively for retrospective searches on chemical information indexed by chemical structures. Before issuing a patent on any discovery, a retrospective search on all previous patents issued must be carried out to determine if the article to be patented is unique. One of the most active fields in patenting is that of chemistry. In order to keep abreast

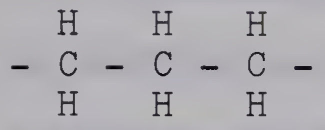
of its responsibilities, the U.S. Patent Office has developed a mechanized approach to the problem. The result is a computer program called HAYSTAQ. No figures are available on the time required to design and implement the HAYSTAQ system.

The HAYSTAQ system depends upon the matching of chemical structures; no other information, such as process or physical properties, is used in indexing the file, although this type of information will be used in future versions. All patents on chemical information are coded by a team of professional chemists. Each indexer analyzes the chemical information, draws the structural diagrams of all chemical compounds used in the claim and then codes these structural diagrams in a form acceptable to the computer. This information is punched on paper tape with a Flexowriter, machine edited, and then coded and compressed onto a magnetic tape file. This file, which uses the coded chemical structures as indexing terms, is used for retrospective searching.

In order to produce a workable system, chemical structures must first be separated into functional groups by the search program. For example, a complex compound such as 3-phenyl propylamine,



can be separated into functional groups of  ,

 and $- NH_2$. Any of these groups can then be used as an index term in a retrospective search. However, the indexer codes the entire structure; the program separates all compounds into their constituent parts.

Every functional group is assigned a unique code. For example, the functional group oxy, of the form = 0, is coded 3COFA, and the functional group thio, of the form X - S - X, is coded 3COC8. By indicating connections between functional groups, in the same way as semantic links and roles are used to connect English words to form a concept, any chemical compound can be uniquely represented. It also can be coded directly from the structual diagram. Thus, a representation of the structure diagrammed previously may be as shown below where the functional codes are arbitrarily assigned by the author.

	From Link	Coded Functional Group	To Link
1.		3CO2A	2(1)
2.	1(1)	3C1C1	3(2)
3.	2(2)	3C72A	

Each link field contains the type of link joining the two functional groups in parentheses.

Another generic extension of the coding scheme used by HAYSTAQ is the Markush function. Instead of specifying a functional group at each link in a structural diagram, a set of functional groups may be specified. For example, a compound may be represented by

$$R_1 - 0 - R_2$$

where

$$\begin{array}{ll} R_1: & NH_3 - \\ & H - 0 - \end{array}$$

and

$$\begin{array}{ll} R_2: & - 0 - H \\ & - H \end{array} .$$

Any combination of R_1 and R_2 functional groups may be used in the original structure. By using this coding scheme, many variations of similar chemical compounds may be represented efficiently.

Requests are coded in the same way as the chemical information. Chemical structures are expanded and coded, and the entire tape file is scanned. Each entry in the file is matched with the request by topologically matching the request and each item in the file. If a complete match

is found, it is output as satisfying the request. However, screening techniques are used to reduce the number of topological matchings. For example, the functional groups are divided into two generic classes. If the more generic of these classes does not contain all the functional groups of the request, no further matching is done; if it does, matching continues.

This system is used exclusively for retrospective searching. Many of the difficulties encountered when dealing with natural language documents are not encountered with this system. A request is either completely satisfied or it is not. Semantic analysis of the meaning of words is not necessary. A well structured and fully defined language is used (the structural formulae of compounds) which has no inherent ambiguities. Hence, conversion to a computer is relatively easy.

4.2 Real-Time, Time-Shared Systems

The second portion of this chapter deals with real-time information retrieval systems designed for time-shared computers. Some authors, for example Licklider (1965) and Swanson (1964), feel that future information storage and retrieval systems will rely heavily on the concept of time-sharing. Future systems will require "immediate" response and direct communication with the stored information. Since,

in most cases, a user is not entirely aware of what he wants, immediate feedback to and from the system will be required in order to narrow down the request to a form manageable for a machine search. All of these requirements imply a need for a real-time system. However, a real-time system cannot be justified economically unless it is shared among several active users. Thus, we have the concept of time-sharing.

Four real-time information retrieval systems will be discussed; CONVERSE, TIP, SMART, and BOLD. SMART and BOLD are experimental systems, of which only the BOLD system is implemented on a time-shared computer.

4.2.1 The CONVERSE System

This system at Lockheed Missile and Space Company (Drew, Summit, Tanaka and Whitely (1965)) is designed around a time-shared computer with a card reader input device and a teletypewriter output unit. The data base consists of two magnetic tape files produced from over 8000 machine readable master catalogue cards containing all the descriptive information of a wide variety of documents. Four people required six weeks to design, implement and test the system after machine readable documents for testing the system were available. This amounts to 120 man-days.

In order to use the system a user selects the group of descriptors best describing the articles desired. A file of punched cards containing all descriptors in the system stands beside the input terminal. There is also a card reader capable of reading a single manually entered card at a time. The desired descriptor cards are selected from this file and arranged, along with control cards, in a deck to be read by the terminal. After reading all of the selected cards, the system outputs on a teletypewriter the documents that satisfy the request. If three or less documents are found, all the bibliographic data are printed. If four to 15 are found, the document reference number is output. If more than 15 are found, a count of the number of documents is output. The user can then refine and rephrase his query and input the request again.

Two magnetic tape files are used for retrieval. The first file consists of all descriptive information appearing as output. The second file, created from the master catalogue cards, is an inverted file, i.e., each descriptor is followed by a list of documents using that descriptor. Descriptors corresponding to all categories are identified, grouped, sorted alphabetically within categories, and manually edited. Non-significant terms are purged, and synonyms are grouped together. The descriptors are then punched on cards and sequenced for future retrieval

requests. This file is searched in response to requests.

Future design improvements include the use of typewriters as input devices, and a "free" vocabulary, i.e., the user is not constrained to the vocabulary of the machine. All user vocabulary will be translated into a form compatible with the system, and a search on the translated terms will take place. Also, attempts will be made to decrease the amount of data to be searched by means of a screening process designed to retrieve a set of documents containing the documents requested.

Although this system is crude in its manner of operation compared to the systems to be described, it is one of the first operational information retrieval systems employing a real-time, time-shared approach.

4.2.2 The Technical Information Project

The Technical Information Project at Massachusetts Institute of Technology (Kessler (1965a)) is one of the few information retrieval systems which is operational on a time-shared machine. It is designed around the experimental Project MAC complex, a real-time, time-shared system with remote teletypewriter input and output units. Figures are not available on the time required to complete the TIP project.

The body of literature to be searched (journals concerned with physics) is relatively limited and of a

technical nature. The journal title, volume number, page number, article title and author(s) of the article, as well as the bibliographic information of the article, such as journal title or source, volume number and page number, are keypunched on to cards. These input data are edited and placed on a disc for access by the computer. No abstracting, reviewing, or editing of the source material is done. Title words and bibliographic data are relied upon for retrieval of the articles. In this way, no costly data preparation is necessary; however, a great deal of the useful information is lost.

The system consists of three parts. The SEARCH command specifies the range of journals to be searched, such as all of the journals, or only the most recent issues of a journal. The FIND command determines what elements are to be searched for, such as author name, a specific word in a title, a particular citation, or the location of the author, e.g., M.I.T. The third command specifies the type of OUTPUT, i.e., store, print, or count the output.

Since each article does not have an elaborate set of index terms associated with it, there are only two useful methods of retrieving relevant material: by specifying words which may appear in the title, or by locating articles which share a common element such as

author or citation. In order to facilitate this, a set of routines under the name SHARE have been written. The most helpful elements which are shared are the citations. This type of sharing is termed "bibliographic coupling". Earlier publications (Kessler (1963a and 1963b)) discuss the use of this criterion to classify documents into related groups. If two documents cite a common reference, it is assumed that they deal with related material. Tests were performed (Kessler (1965b)) which indicate that, in a narrow technical field, bibliographic coupling is a reasonable method of linking documents.

In order to carry out a full search, a preliminary search may be made using a word contained in the title. It returns a few documents which may deal with the desired topic. By then requesting a search for articles which share citations with these documents, most relevant material may be retrieved. However, at least two complete scans of the file of documents are necessary. No attempt is made to classify items to decrease the amount of material searched.

The concept of bibliographic coupling also aids in browsing. By making a preliminary search for a title word the user can limit the amount of material scanned. By then requesting articles that share citations, he can thread his way through most of the material in a narrow field.

Bibliographic coupling is well suited to mechanization. Since each article is assigned a unique numeric name, searches are easily implemented. No problems arise concerning the meaning of words. On the other hand, much of the information that could be coded is lost by retaining only the identifying and bibliographic data. Any paper that deals with a new subject has very few citations to link it to related material. The bibliographic coupling approach appears to be satisfactory for a very narrow field, such as subfields of physics, but it is unsatisfactory for application to a broad spectrum of knowledge.

The Technical Information Project has met with enthusiastic use at M.I.T. Brown (1966) describes a project which is well suited for the system. In order to update his publication Basic Data of Plasma with more recent data, a search is made of all entries which cite relevant articles, and the user is notified. These articles are then reviewed and new data are added to his book.

An area which is just beginning to be explored, and which is a direct result of time-shared computing, is that of using the computer as a communication device. The Selective Dissemination of Information then becomes a minor function of the system. As an article is entered into the system, all user interest profiles are scanned to determine if the article is of interest to them. If so, it is stored

for future printout, perhaps on the terminal itself at the request of the user. Alternatively, a message may be printed in a form suitable for mailing to the user. Other users' files or interest profiles may be scanned to determine whether they are interested in the same field as the searcher. Time-sharing is introducing an entirely new concept into information retrieval and it has a great deal of potential value. However, much research is still needed to determine the best ways of utilizing the new capabilities.

4.2.3 The SMART System

The SMART information retrieval system (Salton (1964) and Salton and Lesk (1965)) is a computerized system which is designed to take full advantage of user interaction with the machine. It is an iterative system which allows the user to specify a search request, analyze the output, and repeatedly respecify his request or the search mode until his needs are fulfilled. It is an experimental system designed for, but not yet implemented on, a time-shared computer. It can also be used as a vehicle for testing various storage, analysis, and search strategies on the full text of documents. The SMART system is a result of an estimated two to three years work. Accurate estimates are difficult since the system is continually evolving.

The system is designed such that, by repeated

specification of many variables, the user retains a great deal of control over the system. After determining which analysis procedures (which will be discussed below) are desired, the user formulates a search request in full English sentences, with no prior coding. His request, along with the full text of all documents in the collection, is analyzed according to the specified analysis procedures, and is correlated with the documents; the highly correlated documents are returned to him. If the results are unsatisfactory, the user can either reformulate his request, or can change the mode of analysis, or both, and rerequest a search. This procedure continues until the user is satisfied.

The analysis system consists of a supervisor, called CHIEF, which can call on various processing subroutines. CHIEF can accept eight input instructions which specify the type of processing to be done and also 35 control options for the various processing instructions. These instructions control the type of analysis to be carried out on the document.

The entire text of the document is input to the system with no prior editing or indexing. Since no processing is done on the input data, the analysis procedures, which have been reviewed (Salton (1963)) and evaluated (Salton (1965)), are the heart of the SMART system. The entire structure of the system depends on these procedures. A request for a

document can be considered as nothing more than another document, to be analyzed by the same procedures used to analyze the text of the documents.

The alphabetic dictionary lookup procedure is an essential step in the analysis and consists of normalizing the vocabulary used in the text. Every word is scanned, and the high frequency low information content words, such as "a", "the", etc. are discarded. The remaining words are separated into stems and affixes by matching against a prestored thesaurus of possible words with corresponding codes. Every stem is replaced by a concept code number, and a syntactic code is assigned for every stem and affix. Variations in word spellings due to addition of affixes, such as pluralizing by changing "y" to "i" and adding "es", are allowed for. If a word cannot be located in the thesaurus a notification is sent to the user, and that word is disregarded in further processing. Synonymous words are assigned the same concept number.

At this point, all words in the text have been analyzed and transformed into a numeric form suitable for mechanical manipulation. The dependency of the subsequent procedures upon vocabulary is thus decreased, and synonyms are matched; also, the meaning of specific words is broadened through synonymous concept numbers.

In the procedures to be described it is sometimes

desirable to retain the original input text words. To facilitate this, a so-called "vacuous" dictionary can be constructed by assigning dummy concept numbers to every new word encountered in the text. Thus, the procedures described above can be used for analysis, and every word in the input text can retain its uniqueness.

An optional method for expanding a document's concept is by consultation of a hierarchy of concepts which can be thought of as replacing a library's classification system. The hierarchy consists of a treelike structure representing relationships between concepts, with each node corresponding to a concept number. Though not stated explicitly, it appears as if the hierarchy is constructed manually. As one moves up the tree, a generically superior (father) concept can be referenced, and if one moves down the tree, a generically inferior (son) concept can be obtained. Those concepts on the same level (brothers) can be accessed, and concepts can be cross referenced, i.e., one can enter at another related node from anywhere in the tree (see Figure 4.2.1). List processing techniques are used to process this tree as well as a tree of concept numbers, which are used as entry points to any node in the tree.

From the alphabetic analysis, we have the concept numbers associated with each sentence. By sorting on concept number, the frequency of occurrence of each concept number,

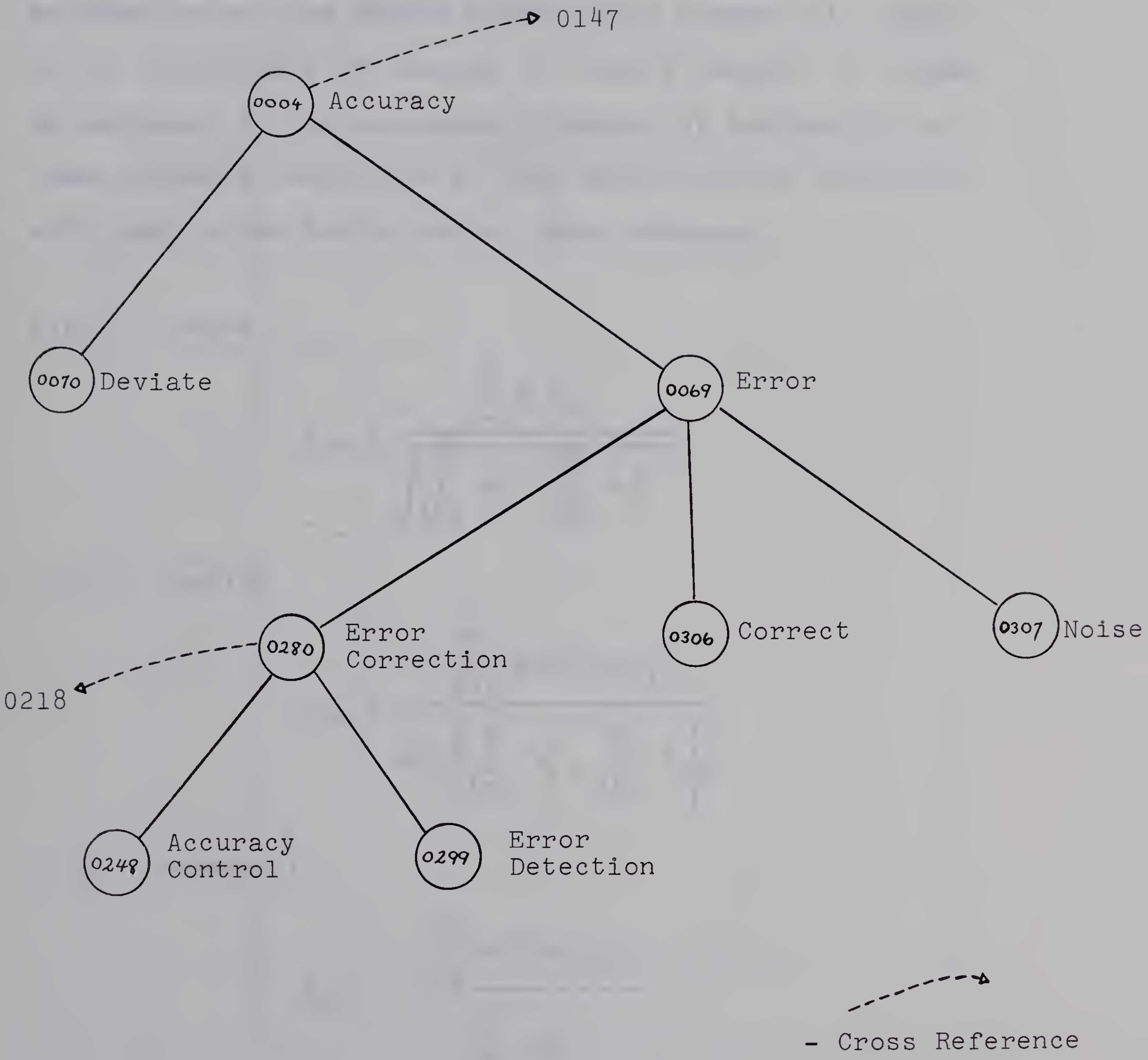


Figure 4.2.1

Concept Hierarchy

[illegible]

[illegible]

along with its corresponding sentence number can be calculated. From this, a concept-sentence incidence matrix can be constructed (see Figure 4.2.2), with element ij equal to n if and only if concept i occurs exactly n times in sentence j. To calculate a measure of similarity between concepts, every row of this matrix can be correlated with every other row by one of three measures:

(4.1) Cosine

$$r_{ab} = \frac{\sum_{i=1}^{m} a_i b_i}{\sqrt{\sum_{i=1}^{m} a_i^2 \sum_{i=1}^{m} b_i^2}}$$

(4.2) Overlap

$$r_{ab} = \frac{\sum_{i=1}^{m} \min(a_i, b_i)}{\min\left\{\sum_{i=1}^{m} a_i, \sum_{i=1}^{m} b_i\right\}}$$

(4.3) Asymmetric

$$r_{ab} = \frac{\sum_{i=1}^{m} \min(a_i, b_i)}{\sum_{i=1}^{m} a_i}$$

$$r_{ba} = \frac{\sum_{i=1}^{m} \min(a_i, b_i)}{\sum_{i=1}^{m} b_i}$$

SENTENCE NUMBER

		1.	2.	3.	- - - -	m
CONCEPT NUMBER	1.	0	3	2		0
	2.	4	1	0		1
	3.	0	0	2		7
	- - - -					
	p	6	2	0		0

Figure 4.2.2

Concept Sentence Matrix

If we compute the correlation of all concept pairs, a concept-concept matrix of correlation measures can be constructed. This measures the strength of association between two concepts within a sentence.

There are two reasons for measuring concurrence over the range of a sentence. A group of concepts which correlates highly within a sentence are, in all probability, a single concept and can be replaced by a single concept number. Also, sentences can be ranked by significance, and only the high ranking sentences need be used for further analysis; or, they can be used to produce an autoabstract, in a manner similar to the approach used by Luhn (1958).

There are two methods of producing a single concept from two or more highly correlated concepts. Firstly, a dictionary of statistical phrases, consisting of concept pairs with no semantic coupling can be consulted, and the new concept number can replace all occurrences of the old concept numbers in each sentence. Secondly, concepts can be clustered. A single concept can be chosen, and a second concept which correlates highly with it can be added to the cluster. A third concept which correlates highly with both terms can be added to the cluster, and so on. This cluster can then be replaced by a single concept number.

Syntactic processing permits a refinement of reduced requests and documents by retaining some of the syntactic relationships of the document. The significant sentences

of both the documents and requests, as described above, may be used for processing. With this mode of analysis, terms or concepts may be clustered if and only if the syntactic relationships are identical. A subroutine has been designed to input the stem and affixes of the significant sentences, and to output the sentences in a syntactic tree form. This tree is then compared to a prestored tree, called the "criterion phrases" dictionary, which contains information about the syntactic relationships between concepts. If a sentence, or parts of a sentence, match the criterion phrases tree, it can be expanded by the addition of further concepts.

Up to this point, documents have been characterized by concept numbers which may have been expanded by hierarchical, syntactical phrase or criterion phrase analysis and by a concept-sentence concurrence matrix. The analysis may be supplemented further by constructing a concept-document matrix, similar to the concept-sentence matrix, except that the entire document, not just sentences, is used. This matrix can be subjected to row correlation methods, as was the concept-sentence matrix, and the concept clusters determined by highly correlated concepts. Any concept in the cluster may then be replaced by a single concept representing the entire cluster.

By using this matrix, and correlating its columns

certain documents and prior requests may be clustered. Documents which then correlate highly with the request vector are returned as satisfying the request. The request vector may have its terms weighted by the user, in addition to being expanded by the above analysis procedures. Thus, the user may retain control over the terms used in the search. The user can vary four variables which affect the search. He can specify the correlation measure used, he can change the number of documents output by changing the correlation threshold value, and he can vary the kinds of documents output either by varying the analysis procedures used or by respecifying the search request.

The SMART system is one of the more advanced experimental information storage and retrieval systems in operation today. Combinations of analysis procedures resulting in relevant documents being returned, and discovery of optimum methods of dealing with man-machine interaction, should result from this experiment.

4.2.4 The BOLD System

The BOLD (Bibliographic On-Line Display) system described by Burnaugh (1966) is a real-time on-line experimental system which is designed as a vehicle for research as well as information storage and retrieval. It is operational on a time-shared machine which uses teletypewriters and cathode ray

tubes (CRT) as terminal devices. The use of CRT devices introduces new possibilities into the information storage and retrieval cycle. Since the time required to output a page of information on the CRT is very small, it can be used as an interrogation device. Upon finding information which is required in hard copy form (printed on paper) for future reference, the information on the CRT can be transferred to the teletypewriters or to magnetic tape for off-line listing. Estimates of development and implementation times are not available for the BOLD system.

The BOLD system is divided into two sections: retrieval and data base generation. The data base generator creates a data base from input data, which can be in almost any form. In the illustration discussed, the data base consists of bibliographic items containing information, referred to as "descriptors", such as title, author, accession number, and the indexing terms of a document. Each entry contains terms subordinate to the descriptors which describe the entry, e.g., *author / Jones, R.T. //. A table of these terms is constructed, with a single address pointing to a data entry using this descriptor. Within this data entry there is an address pointing to another data entry using the same descriptor and so on, until the last data entry points to the first data entry, thus forming a loop. Every document must be manually indexed: the data base generator

is only a convenient method for storing bibliographic items.

In order to transform a natural language request into the language utilized by the data base in the BOLD system, a dictionary of authorized terms is constructed. The basic entity in the dictionary is the descriptor, under which all entries are defined. This dictionary consists of a hierarchical structure that represents a classification of every term in the data base. The user has a high degree of control over this dictionary; he can add, delete and replace terms and change the relationship of terms already existing in the system to suit his own needs. Every user can interrogate the dictionary to learn the words authorized for a search, and he can then form a suitable request. For example, to locate all words of the root HEAT the user would type .HEAT . The system would then return a count of the entries under all root forms of HEAT, and output information similar to the following:

```
6 entries are ref'd by heat
1 entries are ref'd by heaters
2 entries are ref'd by heating.
```

Other interrogation instructions to the dictionary, for example requests for equivalent retrieval terms, can be used. After determining terms which are authorized for the

system, the user may formulate a request by connecting retrieval terms by Boolean operators, e.g.,

.HEATERS and LIGHTING, or

.*AUTHOR = JOHNSON, R. D. and *TITLE = HEATERS.

In order to retrieve documents, a man-machine dialogue is initiated. The system begins by listing all categories of the data base on the CRT. Using a light pen, the user selects a category to which the system responds by flashing all subcategories. The user continues to work his way through a classification tree until retrieval terms which can be used in a search request are presented. He then selects one of two modes, Search or Browse. In the Search mode, a request is formulated by use of terms plus Boolean operators, and all document identifiers satisfying the request, ranked by the number of retrieval terms it contains, is returned. The user may then delete any document, and the next document in the list is presented. The abstract of any document may be requested at any time to determine if the article is, in fact, relevant.

In Browse mode, any descriptor may be used. The browse may be limited to specific categories, or may range over the entire data base. The user can browse his way through documents retrieved by this method, and inspect interesting

abstracts at will. Terms in a request can be deleted, and output can be transferred from the CRT to an off-line device. The dictionary items can be manipulated by adding, deleting, and replacing terms as well as changing relationships between existing terms.

The design of the BOLD system allows use of many dictionaries biased in favor of each user, and all relating to a single data base. Thus, it is possible for a user to design his own basis of communication with the machine by an individual vocabulary and classification scheme.

CHAPTER V

THE SARA SYSTEM

5.0 Introduction

This chapter will describe an information storage and retrieval system developed on, and for, a remote-access, real-time, time-shared computing system. Although the system is intended primarily for applications in the field of history, it could also be applied to other fields. The system, called SARA (Storage And Retrieval Alberta), has been developed with two objectives in mind. First, it should be sufficiently general to be of use in more than a single discipline; second, for reasons of efficiency the use to which the system is to be put is taken into account. For example, a general method for dealing with index terms is used. If an index term has been entered in the thesaurus, it may be used as an index term; there are no restrictions on the length of the term, the meaning of the term, and so on. On the other hand, an efficient and less general method is incorporated into the system because of the dependence of history on dates.

Special attention is paid to storage methods and information manipulation. Peripheral devices are not available because of present restrictions on the programming language. However, an efficient method of storing large masses of data on a random access device is simulated.

The selection of index terms for the documents was carried out in collaboration with the Department of History. Guidelines were laid out to indicate the approach to be taken in selecting the index terms in order that the selection remain compatible with the computer. Appendix A discusses the approach taken and some of the problems encountered.

A brief description of the programming language used and the environment in which the system operates introduces the next section. A description of the SARA system from the user's point of view and of the internal workings of the system concludes the chapter.

5.1 A General Description of the SARA System

The SARA system consists of two sections: the selection and indexing of the documents, and the mechanized storage and retrieval part which manipulates the indexed documents. The first section is under the control of the history participant, and all decisions concerning the choice of index terms, their interrelations and the documents to be indexed are his. The author had little control over these decisions; the only control retained was that concerning compatibility of index terms to the computer. Appendix A gives a more complete description of this part of the SARA system.

The second part of SARA deals with the storage and retrieval of the indexed documents. It is written in a new programming language, called APL (see Iverson (1962) or

Falkoff and Iverson (1966)) for time-shared computer hardware. Following is a full description of the mechanized portion.

5.1.1 The Hardware Environment and the Programming Language

The system is designed around an IBM System 360 Model 67 computer with IBM 2741 remote access terminals. This is a large computing system capable of servicing many users concurrently. Hardware of this type will be vital to the efficient operation of future information storage and retrieval systems. Many users may utilize the hardware by searching a common data base with different search requests, and, hence, share the cost of operation of the system. The BOLD system already described foreshadows such use. The system utilizes only the hardware mentioned above, since no input or output commands exist at present for peripheral devices. All communications between the machine and the user is via the terminals. It is understood that this restriction will be removed shortly. Input and output commands may, for example, refer to magnetic tape or discs. The APL system, used to develop this portion of the SARA system, utilizes the equipment mentioned above, as well as some large capacity random access storage such as disc or drum. Such random access devices will also be necessary for any large scale information retrieval system.

APL is specified so that it is oriented toward on-line communications between man and machine. It is well suited to man-machine communication since its set of operators is very precise and very powerful. This power, however, is restricted primarily to its arithmetical and logical capabilities. For example, arrays can be handled simply and easily. The notation $C \leftarrow A +.\times B$ multiplies two matrices of suitable dimension, *A* and *B*, and stores the result in *C*. Other capabilities necessary for efficient general purpose operation of the computer, such as input and output commands for the use of magnetic tape, disc or cards, are notably lacking.

The programmed portion of the SARA system is divided into three subsystems: the control subsystem, the storage subsystem, and the retrieval subsystem (Figure 5.1.1). These are described separately. Block diagrams and listings of all routines in these subsystems appear in Appendix B, and an example of the use of the system appears in Appendix C.

5.1.2 The Control Subsystem

The control subsystem, *SARACNL*, is a program whose function is to interpret commands and pass control to the appropriate subsystem. Control is returned to the subsystem *SARACNL* upon exit from the storage and retrieval subsystems, and control is ultimately returned to the APL system from this subsystem.

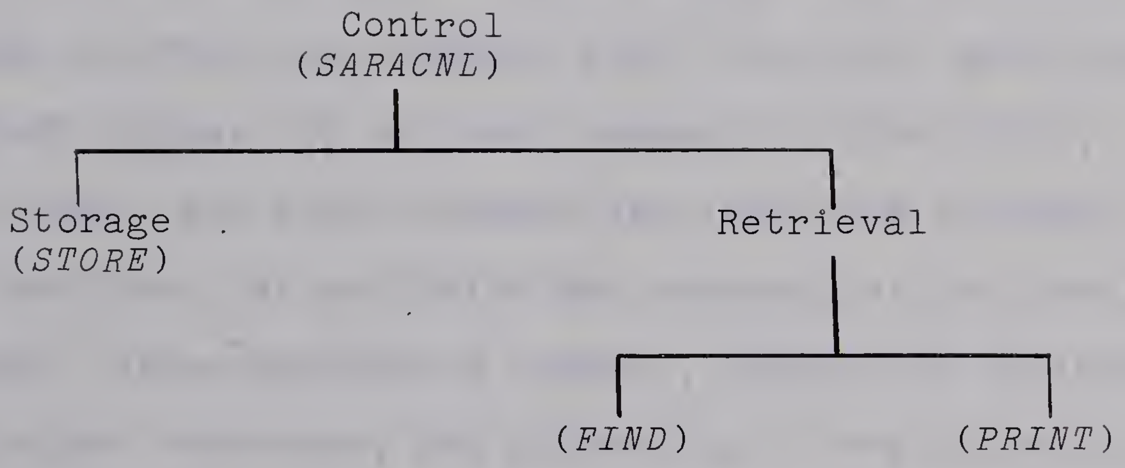


Figure 5.1.1

SARACNL Program Hierarchy

[illegible]

[illegible]

The user signs on to the APL system in the usual manner and requests the control system by typing *SARACNL*. A full description of the operation of the APL system is given by Falkoff and Iverson (1966). In this subsystem, and in all subsystems to be described, the user has the option of obtaining a full explanation of the operation of the subsystem in control immediately upon entering the system. This option may be overridden upon entry to the subsystem by typing *NO* after the command, e.g., *FIND NO*. After typing *GO*, *SARACNL* pauses for an input command: *STORE*, *FIND*, *PRINT* or *END*. The *STORE* command initiates the storage subsystem, and the *FIND* and *PRINT* are commands of the retrieval subsystem. After entering a command, control is relinquished to the proper subsystem, and processing of the function entered begins. *END* passes control back to the APL system.

5.1.3 <u>The Storage Subsystem</u>

The storage subsystem, accessed by typing *STORE* when the main program, *SARACNL*, is in control, stores documents on the simulated peripheral equipment. Where applicable, terms are edited and added to the proper files.

After entering the storage subsystem, the subsystem types a system-assigned document number, followed by *TITLE*. The user enters the title of the document, which is placed in the simulated peripheral file *PERMEM*. *AUTHOR* is then

requested. The author must be input in the form SURNAME-INITIALS. The name is edited. If it has been used previously, the document number is stored in its cell in the matrix *M̈EM*; otherwise, the surname is added to the list of index terms, and a new entry is created in the main file, *M̈EM*. *JOURNAL* is then typed. The input expected is of the form JOURNALNAME, VOL. XX. The name of the journal must be in the list of authorized terms, *L̈ST*; otherwise, the journal is rerequested. The volume number may be omitted if desired. The word *XTERMS* follows. One index term at a time may be entered. Each is edited to determine if it is an authorized term. If it is not then notification is given to the user. Index terms may continue to be entered until the user types *END*. As each term is entered, the current document number is stored in each index term's cell. *YEARS* appears after the user types *END*. The years the document deals with may be entered in one of four formats as indicated by the following examples:

1) 1920
2) 1920,1922
3) 1920,1922,1930
4) 1920-1930.

The document number is entered in the years file, *ŸMEM*. If years are not applicable, any character other than a digit

may be entered and no action will be taken. The final entry for the document is the *ABSTRACT*. If applicable, an abstract may be entered. *MORE?* is typed, and the user replies with *YES* or *NO*; if *NO*, control is returned to *SARACNL*.

The storage subsystem is simple, but it serves the needs of this system. When input and output commands become available to APL, a different storage subsystem will be required. It would be oriented to the peripheral equipment used to store the documents, and to the input and output commands.

5.1.4 The Retrieval Subsystem

The retrieval subsystem has two commands available to it: *FIND* and *PRINT*. These two commands will be discussed individually.

The *FIND* command initiates a subsystem which asks if the request will have weighted terms, to which the user replies *YES* or *NO*. Each index term in the system may have a numeric factor associated with it. This numeric factor, or weight, determines the relative importance of each term. For example, if index terms *ECONOMICS*, *RELIGION* and *POLITICS* have weights 12, 23 and 31, respectively, more importance would be associated with documents indexed by the term *POLITICS* than with the other two terms. Thus, weighting is a method of assigning relative importance to index terms in

a request. For a full explanation of weighting and its use, see Brandhorst (1966). Continuing on with the description of the *FIND* command, the subsystem awaits a request to be entered. This request can consist of index terms, or years, or both, connected by one of the Boolean operators ∨, ∧, ~, <, ≤, =, ≠, >, or ≥. For example, a request may consist of *ECONOMICS* ∨ *RELIGION* ∧(*PER* ≥ 1921). This requests documents dealing with economics or religion on or after the year 1921. The only index term which can be used with a year is *PER* (period). For example, *ECONOMICS* ≥ 1921 is invalid; *PER* ≥ 1921 is not invalid. This request is edited to insure that all terms are valid index terms; if they are not, the user is notified and the request re-entered by the user. If the search terms are to be weighted, the user is notified as each term requires weighting, e.g., *WEIGHTS? - ECONOMICS*; he inputs a weighting factor for each index term, a non-negative integer less than 100, e.g., 62. After all index terms have been weighted, the user is asked to specify the type of expansion required on the request. It may be expanded up (to include more general terms, one for each request term) by typing *GENERAL*; down (to include more specific terms) by typing *SPECIFIC*; across (to include related terms) by typing *RELATED*, or have no expansion at all by typing *NONE*. Figure A.3 in Appendix A depicts, in graphic form, the hierarchy of terms used. At this point,

the system is ready to initiate a search of the data. The user is asked if a search should be initiated. If *YES*, a search on the stored data begins; if *NO*, the user is asked to state his request again. After searching the entire file of documents, the user is notified by a count of the number of documents satisfying the search, and is asked if the document numbers should be listed. If *YES*, five document numbers are typed, and he is asked if more should be output, and so on, until all document numbers are listed. The user is given the option of expanding the same request in another way and re-searching the entire file (by typing *SAME*), or rephrasing his request (by typing *YES*), or exiting the subsystem (by typing *END* or *NO*). If he exits from the subsystem, he can enter the *PRINT* subsystem to inspect the documents retrieved.

The *PRINT* subsystem lists, in natural language, any or all of the descriptive parts of the specified documents. Any set of the title, author, journal, index terms (including the years the document deals with) and the abstract can be inspected. The document to be inspected is identified by its document number. For example, to inspect the title and abstract of document 2613, the user would type

```
2613,TITLE,ABSTRACT  .
```

Since only the first two characters of each option are inspected, the user could also type

2613,TI,AB

and the same information would be output. The system would list the title and abstract, and ask if more documents are to be processed. If the user requires that all parts of the document be typed, he enters 2613,*ALL*. If no more documents are to be processed, the user may exit from the subsystem by typing *END*. He is then in a position to request more documents, or to relinquish control to the APL system by typing *END* again.

The user has a large degree of control over the system. He has options available to him as he formulates his request, as well as a natural language interface with the machine. If a particular request does not satisfy his needs, he can rephrase the request, or he can expand it in one of three ways, i.e., make it more general, more specific or include related terms.

5.2 Details of the Operation of the System

The control subroutine, *SARACNL*, interprets the four commands *FIND*, *PRINT*, *STORE* and *END*, and transfers control to the appropriate subsystem. If the command is not one of these four, the command is re-requested by the system re-typing *GO*. Upon exiting from any one of the three subsystems,

control can be transferred to any other subsystem, or control can be returned to the APL system by typing *END*.

The *FIND* subsystem, a major portion of the retrieval subsystem, will be illustrated by following the steps of transforming a natural language request to reverse Polish numeric notation used to search the file of documents. The original natural language request is transformed three times before it is in reverse Polish numeric notation. The original request is first transformed into a numeric vector, or string; each component of this string corresponds to either an index term or an operator. This numeric string is then expanded to include general, specific or related terms, if requested by the user. The third transformation converts the expanded numeric string to reverse Polish numeric notation. This transformation will be explained in detail later. It is this numeric string that is used in the final step of the retrieval, the search procedure.

Initially, the subsystem asks if the terms are to be weighted by typing *EQUAL WEIGHTS?*; if *YES*, all terms have equal weights of one. It will be assumed in our illustration that the terms will be weighted. The following request is entered.

$$(CHURCH \lor STATE) \land ECONOMICS \land (PER \geq 1650) \land (PER \leq 1900).$$

This requests all documents concerning church or state economics, dealing with the period 1650 to 1900, inclusive. The Boolean operators $\wedge$ (and), $\vee$ (or) and ~ (not), as well as the relational operators (to deal with years) are available to the system.

The open parenthesis is first encoded to its numeric equivalent, -1. The first index term, *CHURCH*, is checked to determine if it is an authorized index term by consulting the thesaurus for valid index terms (*$\ddot{L}$ST*). If it is, a unique non-negative numeric code, called the concept number, corresponding to the index term is determined, and a weight is requested for the term. This weight is combined with the concept number by dividing the weight by 10^6, and adding it to the concept number. Hence, to weight concept number 12 by 68, we would have $12 + (68 / 10^6) = 12.000068$. This is entered in the request vector. The next element, the $\vee$ (or) operator, is coded to its numeric equivalent, -2. Note that all operators and parentheses have numeric codes less than zero. This process continues until the (numeric) years are encountered. These are divided by 10^4 and entered in the coded string as quantities between zero and one. Hence, the year 1921 would have a numeric code of 0.1921. After coding, the sample request would be transformed into the numeric vector

-1, 31.000012, -2, 172.000030, -11, -3, 61.000014,
-3, -1, 1.000001, -9, 0.165001, -11, -3, -1, 1.000001,
-6, 0.190001, -11

with the following concept numbers and weights: *CHURCH* = 31, weight = 12; *STATE* = 172, weight = 30; *ECONOMICS* = 61, weight = 14; and *PER* = 1, weight = 1. All operators (including parentheses) are coded as negative integers, all numeric quantities, such as years, as fractions and all index terms as positive integers. Thus, three classes of possible input can be distinguished in coded form by the range into which they fall. The weights are added to each concept number as fractions.

At this point, the user is asked for the type of expansion required for the request. One of these four commands must be input: *GENERAL*, *SPECIFIC*, *RELATED* or *NONE*. The request is expanded to include the relevant terms in an "or" relationship.

A hierarchy, depicted in Figure 5.1.2, is represented in the computer by a three-dimensional binary array, called *T̈REE*. This hierarchy is set up manually, and is manually entered in the computer. For the purposes of the present discussion, *T̈REE* will also be represented by two matrices named *A* and *B*. Both *A* and *B* are square matrices, with the number of rows and columns equal to the number of index

terms in the hierarchy. In this example, there are eleven rows and eleven columns. If index term j implies index term i, e.g., *HISTORY* implies *CHURCH*, element ij of either matrix *A* or *B* contains a one. Otherwise, it contains a zero. If both elements ij and ji contain a one, index term i implies index term j, and index term j implies index term i (see *CHURCH* and *TAXES* in Figure 5.1.2). The hierarchy in Figure 5.1.2 depicts two types of relationships; solid lines represent the direct relationship between terms and the dotted lines represent cross reference index terms. Thus, we have two matrices, one corresponding to each type of relationship. Matrix *A* depicts the direct relationships and matrix *B* depicts the cross references.

Associated with the array *T̈REE* is a pointer vector, *T̈PT*. *T̈PT* has the same number of elements as there are index terms in the system. Each index term has a concept number which is used to subscript *T̈PT*. The corresponding element in *T̈PT* gives the row or column corresponding to that term. If the element is zero, the index term does not appear in the hierarchy. Figure 5.1.3 shows the relationship between concept number, *T̈PT* and *T̈REE*.

Each request can be expanded in one of three ways (besides having no expansion done on it). In order to include more general terms in the request (*GENERAL*), the

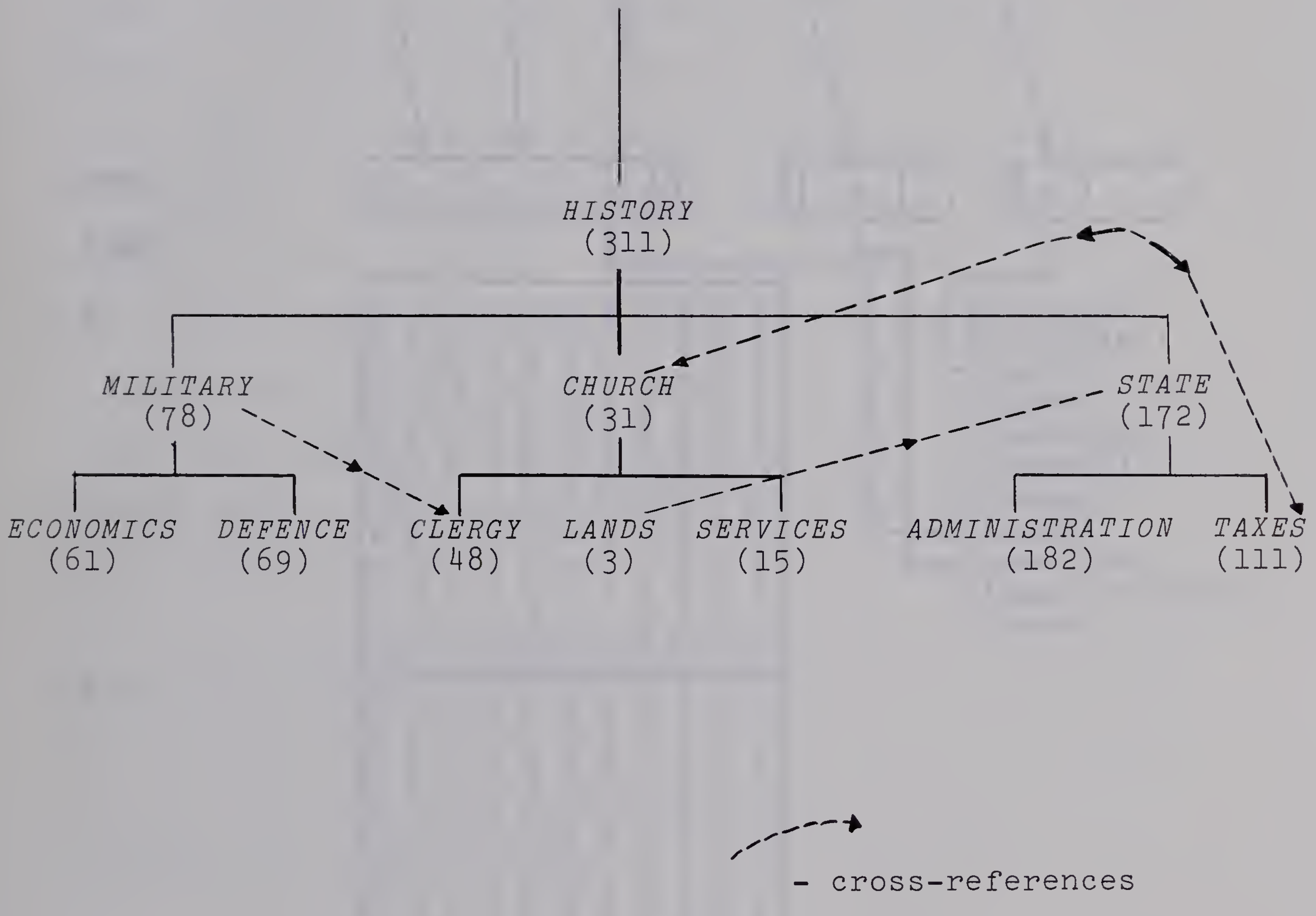
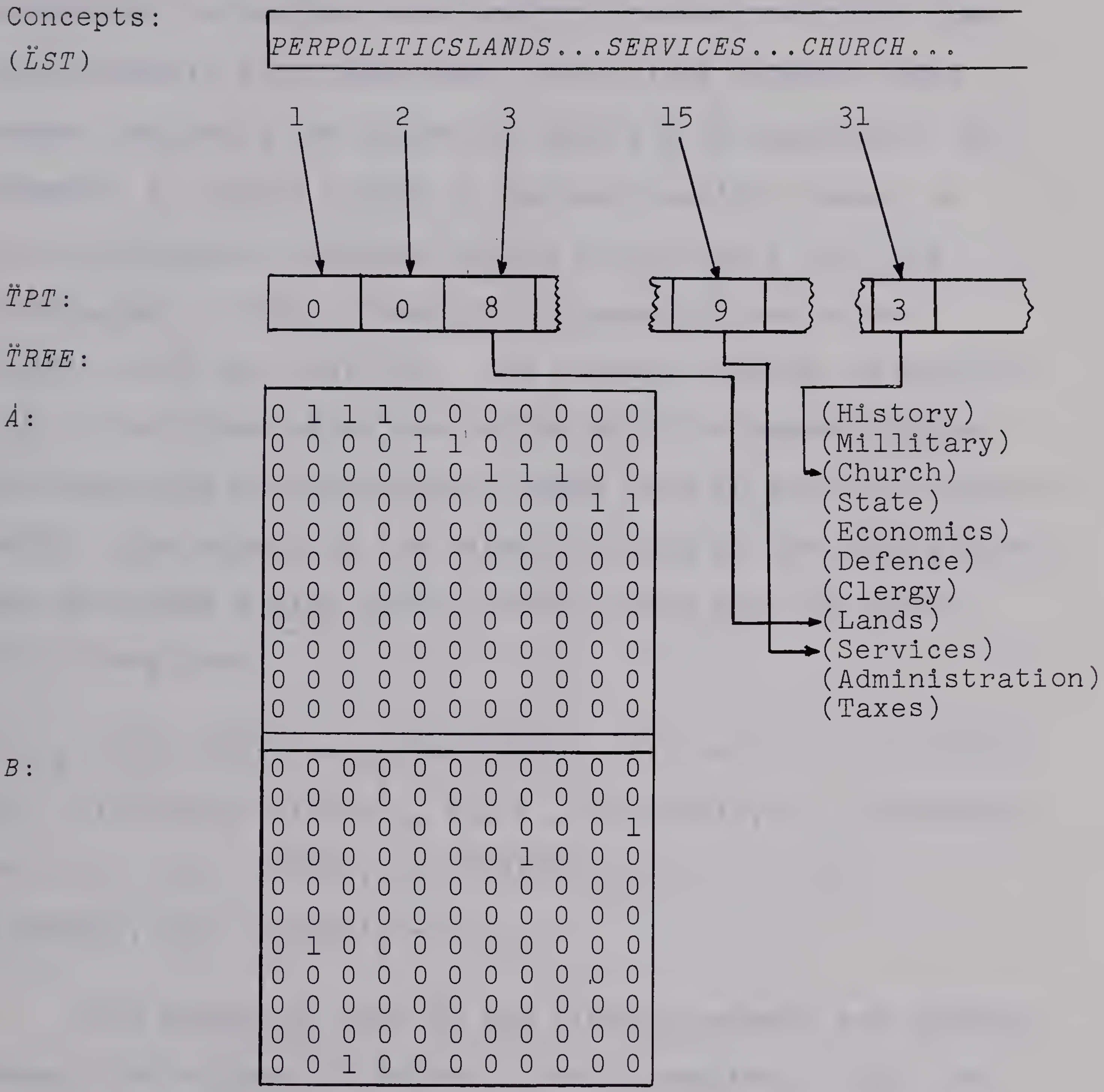


Figure 5.1.2

Expansion Tree

[illegible]

[illegible]



$$\dot{A}: \begin{array}{ccccccccccc}
0&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0
\end{array}$$

$$B: \begin{array}{ccccccccccc}
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0
\end{array}$$

Figure 5.1.3

The Relationship Between Concept Numbers, *$\ddot{T}PT$* and *$\ddot{T}REE$*

column of matrix *A* corresponding to the index terms is consulted; to include more specific terms (*SPECIFIC*), the row of matrix *A* is consulted. To include related index terms (*RELATED*), the column of matrix *B* is consulted. For example, to expand *CHURCH* to include specific terms, we note that matrix elements *A*[3;7], *A*[3;8] and *A*[3;9] are referenced. These elements correspond to the terms *CLERGY*, *LAND* and *SERVICES*. The concept numbers corresponding to the index terms are merged with the concept number corresponding to the original index term in an "or" relationship. Upon expanding the example string to the more general, the following string would result if the tree of Figure 5.1.2 were used:

-1, -1, 31.000012, -2, 311.000012, -11, -2, -1, 172.000030, -2, 311.000030, -11, -11, -3, -1, 61.000014, -2, 78.000014, -11, -3, -1, 1.000001, -9, 0.165001, -11, -3, -1, 1.000001, -6, 0.190001, -11.

The expansion done on the example request was *GENERAL*. Hence, the columns of matrix *A* were consulted. This corresponds to tracing a path one level up the heavy line in the hierarchy of Figure 5.1.2. If, on the other hand, the expansion had been *SPECIFIC*, the rows of matrix *A* would be consulted, corresponding to tracing a path one level lower on the heavy lines of the hierarchy. If the expansion had

been *RELATED*, the columns of matrix *B* would have been consulted. This would correspond to following the dotted lines in the hierarchy. Returning to our example, the numeric string is a coded form of the natural language request

$$((\mathit{CHURCH} \vee \mathit{HISTORY}) \vee (\mathit{STATE} \vee \mathit{HISTORY})) \wedge (\mathit{ECONOMICS} \vee \mathit{MILITARY}) \wedge (\mathit{PER} \geq 1650) \wedge (\mathit{PER} \leq 1900).$$

The numeric string is converted to early operator reverse Polish notation. Lukasiewicz, a Polish logician, demonstrated that if operators were written after their operands, instead of between them, there is never a need for parentheses to show association between the terms. The resulting reverse Polish notation is amenable to searching techniques. For example, the request

$$(\mathit{ECONOMICS} \vee (\mathit{RELIGION})) \wedge (\mathit{PER} \geq 1921)$$

has a reverse Polish representation as follows:

$$\mathit{ECONOMICS},\ \mathit{RELIGION},\ \vee,\ \mathit{PER},\ 1921,\ \geq,\ \wedge.$$

Hamblin (1962) gives a full description of possible Polish notations, and the rules by which a string is transformed to a particular Polish notation. Returning to our example with the numeric string, its reverse Polish notation is

31.000012, 311.000012, -2, 172.000030, 311.000030,
-2, -2, 61.000014, 78.000014, -2, -3, 1.000001,
0.165001, -9, -3, 1.000001, 0.190001, -6, -3.

The corresponding natural language request is

CHURCH, *HISTORY*, ∨, *STATE*, *HISTORY*, ∨, ∨, *ECONOMICS*,
MILITARY, ∨, ∧, *PER*, 1650, ≥, ∧, *PER*, 1900, ≤, ∧.

At this point, the request has been edited, converted to a numeric string, weighted and further transformed into reverse Polish notation and is ready for the search. The user is asked if a search should begin. If *YES*, the matching of each entry in the file with the coded request vector takes place. Document numbers satisfying the request are saved in a vector *ẄV*, and the number of documents satisfying the request is output to the user as *COUNT* = *XX*. Document numbers, sorted on the number of index terms satisfying the request and weights given to each index term, may then be listed.

The information in the system consists of data files and pointer vectors used to locate entries in the data files. Three files, with their corresponding pointer vectors, form the basis of the system. The first file, the thesaurus (see Figure 5.1.4), contains the authorized index terms (*L̈ST*). The concept number of each term is directly related to the

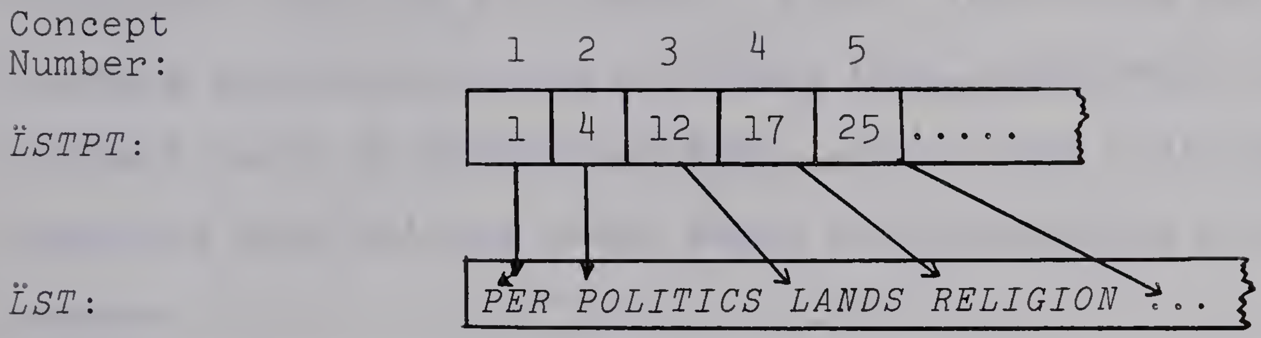


Figure 5.1.4

Thesaurus

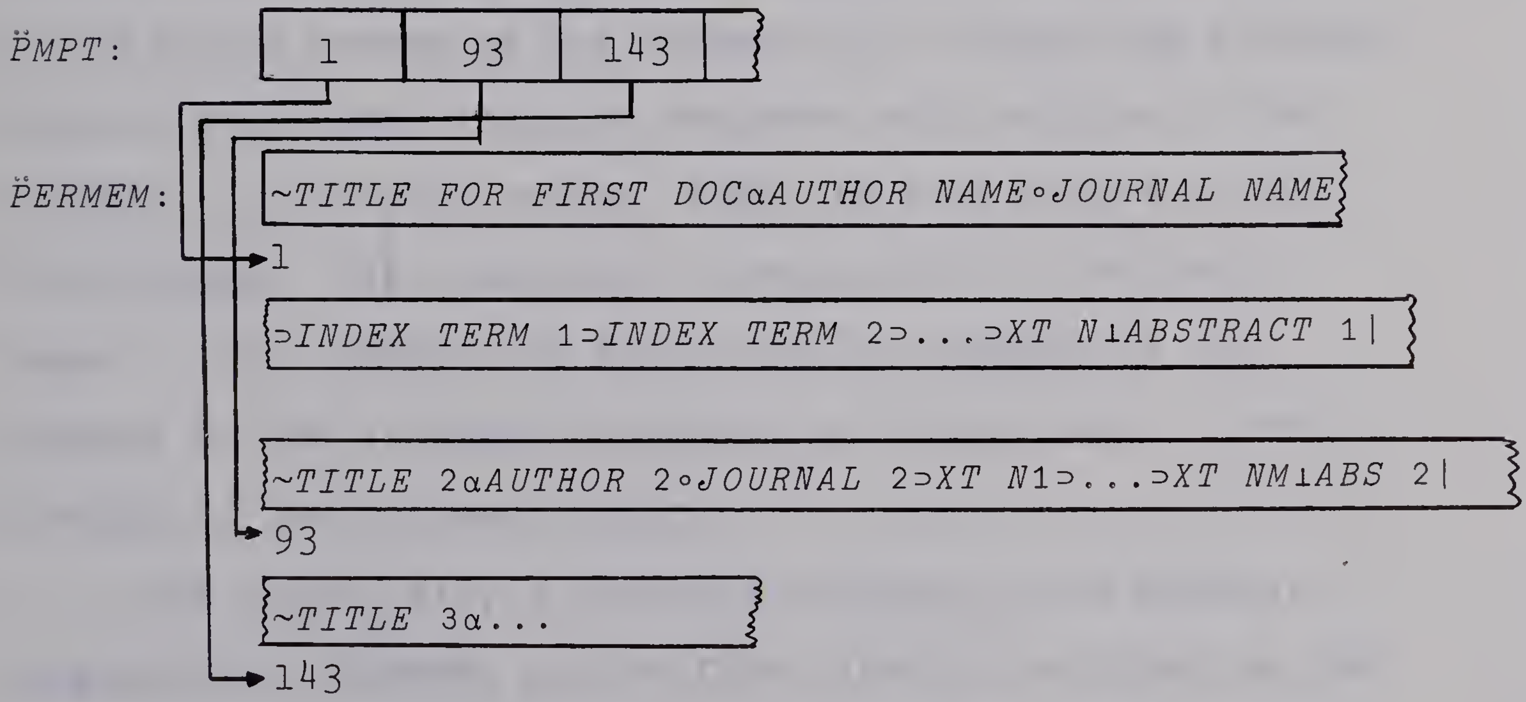
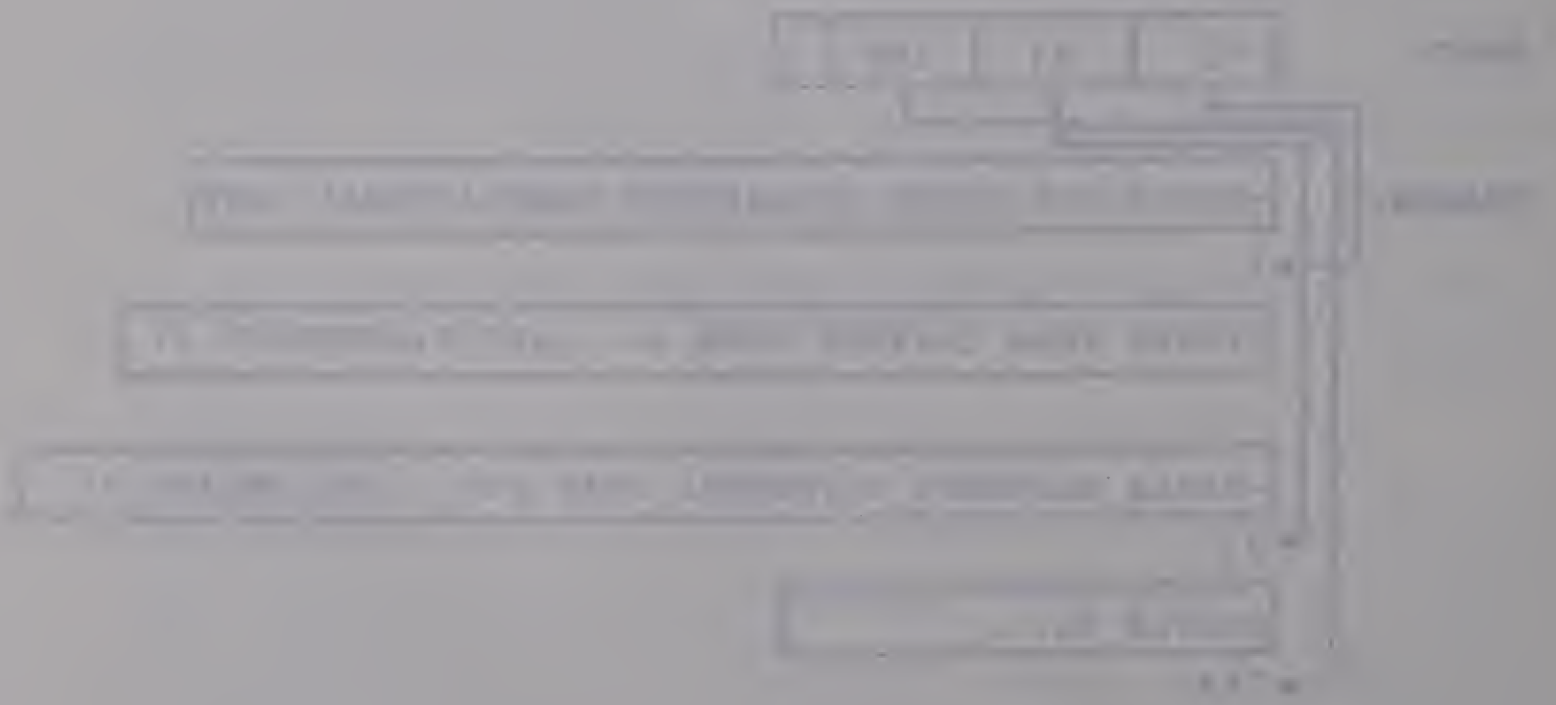

P̈MPT:
1
93
143
P̈ERMEM:
~TITLE FOR FIRST DOCαAUTHOR NAME∘JOURNAL NAME
1
⊃INDEX TERM 1⊃INDEX TERM 2⊃...⊃XT N⊥ABSTRACT 1|
~TITLE 2αAUTHOR 2∘JOURNAL 2⊃XT N1⊃...⊃XT NM⊥ABS 2|
93
~TITLE 3α...
143

Figure 5.1.5

Peripheral Memory Simulation

[illegible]

[illegible]



[illegible]

[illegible]

position in the file that the term occupies; the first term has concept number one, the second concept number two, etc. Associated with *L̈ST* is a vector, *L̈STPT*, indicating the starting position of each new index term, comparable to the approach taken by *P̈ERMEM* and *P̈MPT*. It is this file that is consulted when editing index terms and determining concept numbers.

The second file, *P̈ERMEM*, (see Figure 5.1.5) consists of all input data as they are entered by the user. All alphabetic information which will subsequently be output in response to the *PRINT* command is contained in this file. For example, the title, author, journal, index terms and abstract, all in natural language form are stored on simulated peripheral equipment; the address of the document in this file serves as the document for processing purposes. Special characters serve to separate each section of the document. A pointer vector, *P̈MPT*, is also stored in the file *P̈ERMEM*. Each component corresponds to a document number. The contents of *P̈MPT* at this location is the address of the starting character of the document corresponding to the document number.

The third file, a codified version of the natural language file *P̈ERMEM*, is the file directly utilized by the computer; all operations of the computer are done on this file. It consists of two subfiles: the main subfile, *M̈EM*,

and the years subfile, *ŸMEM*. *M̈EM* is an inverted file, with every index term (excluding years), every author and every journal in the system occurring as an entry. Under each of these entries there is a list of document numbers (or document addresses), each number identifying a document indexed by this specific index term. For example, if concept number 76 was used as an index term for documents 173, 111 and 683, and concept 77 was used as an index term for document 731, part of the main file would appear as in Figure 5.1.6.

The main subfile consists of *M̈EM* and *P̈T*; *M̈EM* is variable in length. Each concept has two positions in which to record document numbers. A third position contains an address pointing to the row containing the next two document numbers corresponding to the same concept number. If there are no more documents, the third column contains the special numeric code 999999. Hence, as more documents are indexed under a single concept number, the file grows in size. However, it is not wasteful of space, since there is at most one vacancy under any concept number. In order to retrieve all documents indexed by a specific term, the document numbers are retrieved by chaining through the file until the terminating numeric code, 999999, is encountered. As the file grows in size, and as experience accumulates, the file can be revised to minimize the number of zero

elements. Thus, the variable $\ddot{N}C$ determines the number of columns in $\ddot{M}EM$. Entry to the file is made by the pointer vector, $\ddot{P}T$. The index term's concept number is used as a subscript on $\ddot{P}T$; the contents of $\ddot{P}T$ at this location is the first row in $\ddot{M}EM$ corresponding to this concept number (see Figure 5.1.6).

The second subfile, $\ddot{Y}MEM$, contains the period that each document covers. Special treatment is given to the problem of periods, or dates, dealt with by the document. It would be impractical, but possible, to use every year from zero to 1967 as a separate index term. Instead the following method was used: each document which has years as one or more of its index terms is entered in a table as pictured in Figure 5.1.7. The first column contains the document number, the second column the starting date, the third either a date or a zero depending on whether the document deals with two specific dates or a range of years, and the fourth contains a date. By allowing a range of years to be used by inserting a zero in the third column, much repetition of dates is avoided.

Presently, no files are on peripheral devices such as disc or magnetic tape. All are simulated in memory by means of matrices and vectors, and remain in memory throughout processing. This is due to the lack of input and output commands of the programming language which will be rectified, it is hoped, in the near future.

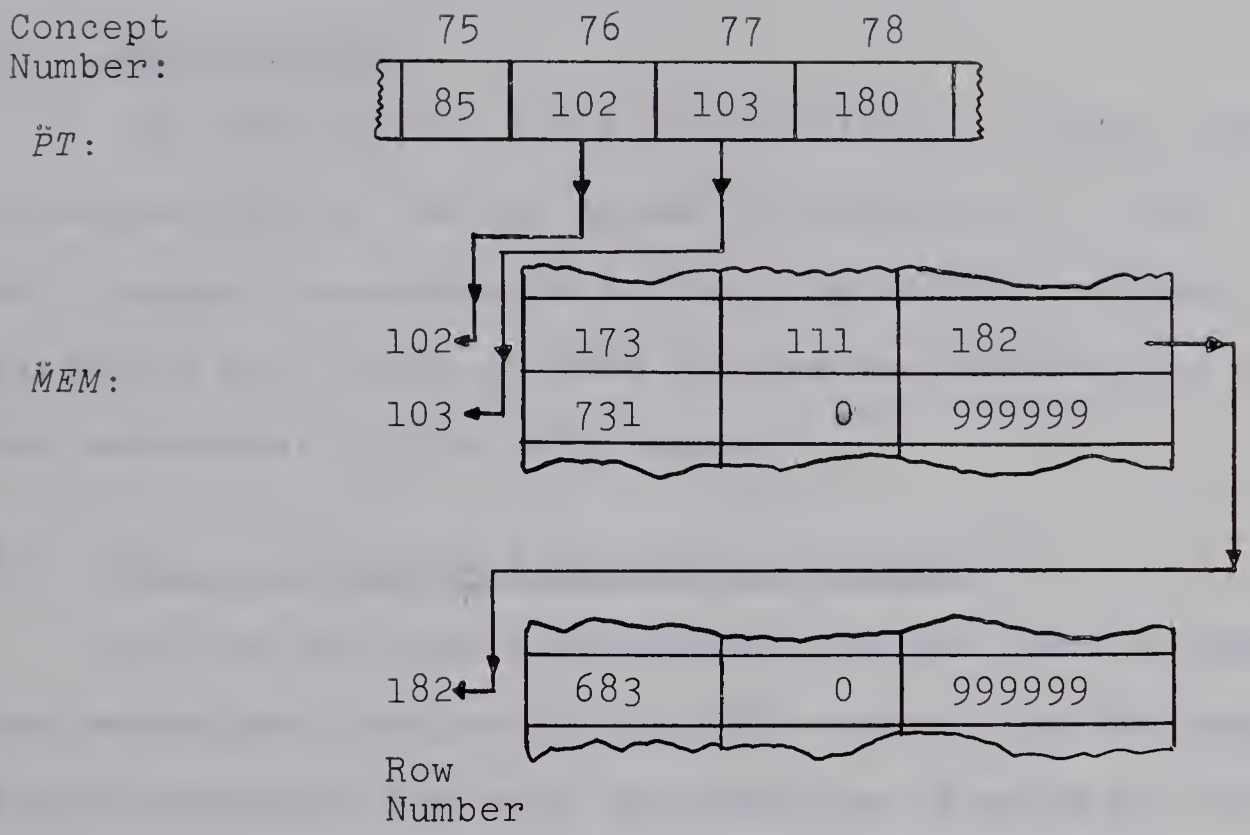
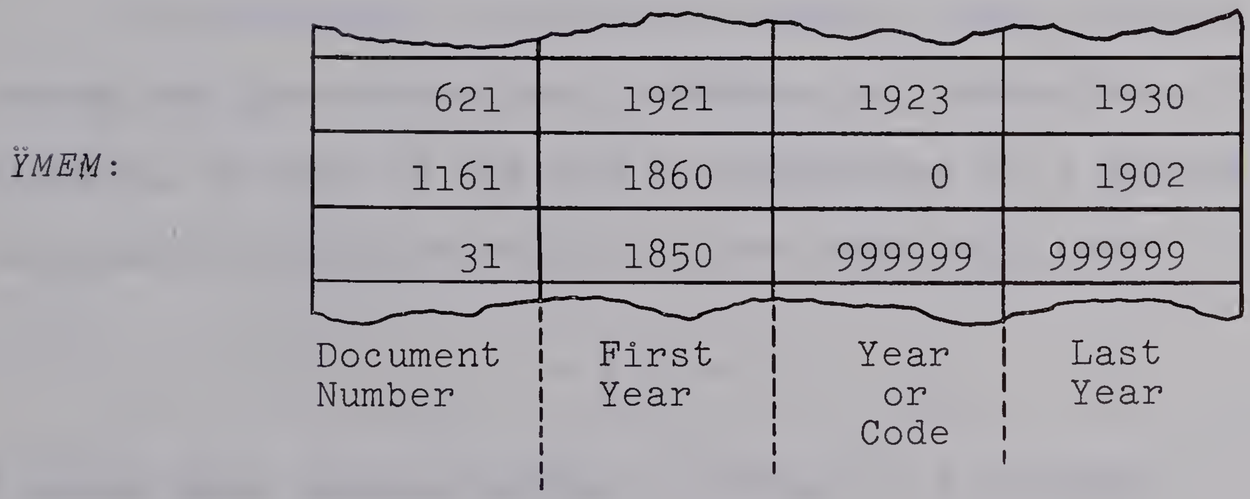


Figure 5.1.6

Main Subfile



Document Number	First Year	Year or Code	Last Year
621	1921	1923	1930
1161	1860	0	1902
31	1850	999999	999999

Figure 5.1.7

Years Subfile

[illegible]

[illegible]

CHAPTER VI

COMPARISON AND EVALUATION OF THE SARA SYSTEM

6.0 Introduction

The SARA system has been described in detail, and illustrations of its use appear in Appendix C. This chapter will suggest improvements to the programming language, compare SARA with other on-line systems and discuss the strengths and weaknesses of the SARA system.

6.1 APL As a General Programming Language

APL is the only programming language used to develop the mechanized portion of the SARA system. As the application encompasses not only the handling of matrices (in peripheral equipment simulation) but also much general data processing, e.g., character handling and input and output considerations, an opinion could be formed as to the suitability of the language as a general processing language.

The language is extremely powerful when dealing with arrays and performing most mathematical operations. For example, to test if the sum of components of a matrix A is equal to a scalar value, X, one need only type

$$Y \leftarrow X = +/+/A \ .$$

Y would then contain either 1 (true) or 0 (false).

However, other aspects of the language could be improved. This investigation indicated several ways in which the language could be modified so that programs for information storage and retrieval could have been written more conveniently. These are as follows:

a) When dealing with functions, the user does not have the power to declare any function parameter as local or global, i.e., call-by-value or call-by-name. All variables are assumed local. Hence, if more than one variable containing a large number of elements is to be operated upon in exactly the same manner, separate functions must be written to handle each variable, and each function cannot have the variable as a parameter. This arises since a temporary copy is made of a variable appearing as a function parameter while the variable is being operated on in the function. If the variable contains a sufficient number of elements to overflow memory capacity if duplicated, it cannot be used as a function parameter. Such variables often occur in information retrieval programs, e.g., thesaurus lists, or lists of document numbers. By giving the user the power to declare a variable in a function header as global (and thus avoid duplication), this problem could be avoided.

b) At present, a function is limited to two parameters which may be scalars or arrays of any dimension.

It would be convenient if functions could be extended to include a larger number of parameters.

c) List processing techniques have many applications, particularly in information retrieval. Commands which would enable data to be processed as lists would also be useful.

d) Looping is necessary for dealing with iterative procedures in functions. At present, no convenient looping command comparable to the DO-statement in Fortran exists for APL. An instruction which would enable the starting value, increment, ending value, variable incremented and end of the loop to be defined would be useful.

e) At present, APL is only available as an interpreter. No object code is generated and hence much work must be duplicated in interpreting each statement each time through a loop. If programs could be debugged in interpretive mode, and subsequently compiled into efficient object code for later production runs, the language would be more efficient.

f) One of the largest bottlenecks in the language at present is its inability to communicate with peripheral devices other than the user's terminal. There are no input or output commands to peripheral devices. If one could direct the system, from the terminal, to input from a large storage capacity device, or to output data to peripheral devices such as printer or magnetic tape, the language would become very attractive to the general computer user and also

the information retrieval programmer. Implicit in this proposal is the ability to format the data as desired by the user.

On the whole, APL is an improvement over existing languages when used for communicating with the computer directly. It is very powerful when dealing with arrays or arithmetical applications. This power, unfortunately, does not extend to features required to make it a more general purpose language.

6.2 SARA and Other On-Line Systems

Other on-line information systems have been developed. SARA is compared and contrasted to four systems discussed previously: CONVERSE, TIP, SMART and BOLD.

The CONVERSE system does not give the user much control over the formulation of requests or output format. For example, the type of output given to the user after completing a search is dependent on the number of matches made; also, any expansion of requests must be done manually. The SARA system allows for more options and control than the CONVERSE system, such as automatic expansion upon request.

The Technical Information Project uses a computer with capabilities comparable to the computer used by the SARA system; it has remote consoles connected directly to a time-shared computer. However, the approach taken by TIP is

unlike that taken by SARA. Where SARA requires the manual indexing of documents prior to storage in the computer, TIP uses only the title, author, source journal and bibliographic data of each document. Little professional manual effort is required to prepare documents for the TIP system. The search strategies then vary according to the differences in the type of data stored. TIP users chain from one document to the next via the bibliographic data, while SARA users retrieve documents by matching index terms in the request and documents.

The SMART system, in part, resembles SARA. In general, the SMART system has greater capabilities than the SARA system. It accepts the full text of a document as input, and automatically indexes the document. Searches are then made on these indexed documents; the user retains a great deal of control over the expansion of the request and output format. SARA requires the manual indexing of the documents, but provides a search procedure resembling that of the SMART system. The user has options available; he retains control over the formulation of the request, the subsequent search and the output.

The BOLD system requires that documents be manually indexed prior to their storage, and requires a list of authorized index terms, similar to SARA. However, in order to search for documents in the BOLD system, the user chains through the hierarchy of authorized index terms. The terms

at the end of the tree (or any level prior to the end) are used as index terms, and the user initiates a search with these terms. Matches are output. SARA, on the other hand, does not require the user to follow through the hierarchy of terms; any index term connected with any other by Boolean operators is valid.

6.3 Strengths and Weaknesses of the SARA System

The mechanized portion of the SARA system has many favourable features. Since it is programmed for a time-shared computer, the availability of the system is restricted only by availability of the computer and typewriter consoles. The system requires little effort on the part of the user to learn its use. Response to queries is almost instantaneous. If an error is detected in a request, e.g., a misspelled index term, the user is notified immediately and the request can be corrected. If a request does not retrieve a sufficient number of appropriate documents, it may be expanded and hence increase the number of documents retrieved. Full English words are used to communicate with SARA; no codes need be remembered. For the user familar with the system, abbreviations of the full words may be used, e.g., YES may be abbreviated to Y. The SARA system has options available, and is easy to use. However, there is room for improvement.

Only a portion of the information storage and retrieval cycle is automated by SARA; manual indexing and abstracting is still required. The SARA system could be expanded to include the mechanization of these processes. The present system is limited to upper case characters only. By modifying some routines slightly, and by using an upper and lower case typeball on the typewriter, this limitation can be overcome. As the number of entries in the main file increases, the response time to the search request will increase. Also, with the incorporation of input and output commands, the system could handle an increased number of documents. However, the capabilities of SARA, like the capabilities of other mechanized information systems such as SMART, TIP, CONVERSE or MEDLARS, are still relatively elementary when compared to human capabilities. Most mechanized systems rely on Boolean operators only; human capabilities such as recognizing near-synonymous words, are not incorporated into the mechanized systems. Further research could be undertaken to improve and extend the SARA system. It is hoped that this project will be continued.

BIBLIOGRAPHY

American Bibliographic Center, 1965. Guidance Booklet, America: History and Life, Clio Press, Santa Barbara, Calif.

Baxendale, P.B., 1958. "Machine-made index for technical literature", IBM Jour. Res. Dev., 2:354-361.

Becker, J. and R.M. Hayes, 1963. Information Storage and Retrieval: Tools, Elements, Theories, J. Wiley and Sons, New York.

Bledsoe, W.W. and I. Browning, 1959. "Pattern recognition and reading by machine", 1959 Proc. of the East. Joint Comp. Conf., 16:225-232. Also in: L. Uhr, (1966).

Borko, H. and M. Bernick, 1963. "Automatic document classification", Jour. Assoc. Comp. Mach., 10:151-162.

Borko, H. and M. Bernick, 1964. "Automatic document classification: Part II, additional experiments", Jour. Assoc. Comp. Mach., 11:138-151.

Bourne, C.P., 1963. Methods of Information Handling, J. Wiley and Sons, New York.

Brain, A.E., G.E. Forsen, N.J. Nilsson and C.A. Rosen, 1962. "Learning machines", International Science and Technology, 658-669.

Brandhorst, W.T. and P.F. Eckert, 1966. Guide to Processing, Storage and Retrieval of Bibliographic Information at the NASA Scientific and Technical Information Facility, Documentation Incorporated, College Park, Md.

Brown, S.C., 1966. "A bibliographic search by computer", Phys. Today, 19:59-64.

Burnaugh, H.P., 1966. The BOLD (Bibliographic On-Line Display) System, System Development Corp., Santa Monica, Calif.

Calingaert, P., 1968. Introduction to APL, Science Research Associates, Inc., Chicago.

Damereau, F.J., 1965. "An experiment in automatic indexing", Amer. Doc., 16:283-289.

Datatrol Corporation, 1965. Final Report on Phase I - Systems Design and Action Plan for the Pesticides Information Center, National Agricultural Library, Washington, D.C.

Doyle, L., 1965. "Is automatic classification a reasonable application of statistical analysis to text?", Jour. Assoc. Comp. Mach., 12:473-489.

Drew, D.L., R.K. Summit, R.I. Tanada and R.B. Whitely, 1966. "An on-line technical library reference retrieval system", Amer. Doc., 17:3-7.

Edmundson, G.P. and R.E. Wyllys, 1961. "Automatic abstracting and indexing - a survey and recommendations", Comm. Assoc. Comp. Mach., 4:226-234.

Falkoff, A.D. and K.E. Iverson, 1966. APL \ 360, International Business Machines Corp., Yorktown Heights, New York.

Feigenbaum, E. and J. Feldman, 1963. Computers and Thought, McGraw-Hill, New York.

General Electric Company, 1963. The MEDLARS Story at the National Library of Medicine, U.S. Department of Health, Education and Welfare Public Health Service, Washington, D.C.

Grimsdale, R.L., F.H. Sumner, C.J. Tunis and T. Kilburn, 1959. "A system for the automatic recognition of patterns", Proc. Inst. of Elec. Eng., 106B:210-221. Also in: L. Uhr, (1966).

Gyr, J.W., J.S. Brown, R. Willey and A. Zivian, 1966. "Computer simulation and psychological theories of perception", Psych. Bull., 65:174-192.

Hamblin, C.L., 1962. "Translation to and from Polish notation", Comp. Jour., 5:210-213.

Iverson, K.E., 1962. A Programming Language, J. Wiley and Sons, New York.

Kent, A., 1962. Textbook of Mechanized Information Retrieval, Interscience Publishers, New York.

Kessler, M.M., 1963a. "An experimental study of bibliographic coupling between technical papers", IEEE Trans. PTGIT, IT-9:49-50.

Kessler, M.M., 1963b. "Bibliographic coupling between scientific papers", Amer. Doc., 14:10-25.

Kessler, M.M., 1965a. "The MIT technical information project", Phys. Today, 18:28-36.

Kessler, M.M., 1965b. "Comparison of results of bibliographic coupling and analytic subject indexing", Amer. Doc., 16:223-233.

Licklider, J.C.R., 1965. Libraries of the Future, The M.I.T. Press, Cambridge, Mass.

Luhn, H.P., 1958. "The automatic creation of literature abstracts", IBM Jour. Res. Dev., 2:159-165.

Marden, E.C., 1965. HAYSTAQ, A Mechanized System For Searching Chemical Information, National Bureau of Standards Technical Note 264, Washington, D.C.

Maron, M.E., 1961. "Automatic indexing: an experimental enquiry", Jour. Assoc. Comp. Mach., 8:407-417.

Minsky, M., 1961. "Steps toward artificial intelligence", Proc. IRE, 49:8-13. Also in: E. Feigenbaum and J. Feldman, (1963).

Nilsson, N.J., 1965. Learning Machines, McGraw-Hill, New York.

Prather, R.C., and L.M. Uhr, 1964. "Discovery and learning techniques for pattern recognition", Proc. Assoc. Comp. Mach. 19th National Conf., Philadelphia, Pa., P-64:D2.2-1 - D2.2-10.

Roberts, L.G., 1960. "Pattern recognition with an adaptive framework", IRE, 1960 International Convention Record, 2:66-70. Also in: L. Uhr, (1966).

Rosenblatt, F., 1960. "Perceptron simulation experiments", Proc. IRE, 48:301-309.

Salton, G., 1963. "Some hierarchial models for automatic document retrieval", Amer. Doc., 14:213-222.

Salton, G., 1964. "A document retrieval system for man-machine interaction", Proc. Assoc. Comp. Mach. 19th National Conf., P-64:L2.3-1 - L2.3-20.

Salton, G., 1965. "The evaluation of automatic retrieval procedures - selected test results using the SMART system", Amer. Doc., 16:209-222.

Salton, G. and M.E. Lesk, 1965. "The SMART automatic document retrieval system - an illustration", Comm. Assoc. Comp. Mach., 8:391-398.

"Schizophrenia reading made easy", 1966. Journal of Data Management, October.

Selfridge, O.G. and U. Neisser, 1960. "Pattern recognition by machine", Scientific American, 203:60-68. Also in: E. Feigenbaum and J. Feldman, (1963).

Silva, G. and C.J. Bellamy, 1965. Language Data Processing - Concordance Generation, Monash University Computer Center Publication R. 2, Melbourne, Australia.

Swanson, D.R., 1964. "Design requirements for a future library", Libraries and Automation, Library of Congress, Washington, D.C.

Tasman, P., 1957. "Literary data processing", IBM Jour. Res. Dev., 1:249-256.

Uhr, L., 1963. "Pattern recognition computers as models for form perception", Psych. Bull., 60:40-73.

Uhr, L., 1966. Pattern Recognition, J. Wiley and Sons, New York.

Vickery, B.C., 1965. On Retrieval System Theory, (Second Edition), Butterworths, London.

Walston, C.E., 1965. "Information retrieval", Advances In Computers, 6:1-30.

APPENDIX A

SELECTION AND INDEXING OF DOCUMENTS

The documents used to test the SARA system consist of manually indexed papers appearing in the two history journals Agricultural History and Saskatchewan History. The papers were indexed by a Graduate Research Assistant in the Department of History at the University of Alberta in collaboration with the author. The history participant chose all articles to be indexed, all index terms and the relationship between index terms. The author's role was primarily to maintain sufficient control over choice of index terms and the indexing so that they would be compatible with the computer. Much assistance was willingly given by everyone concerned. No extensive user studies have been carried out on the SARA system. Figure A.1 depicts, in block diagram form, the general flow of the statement and solution of the problem.

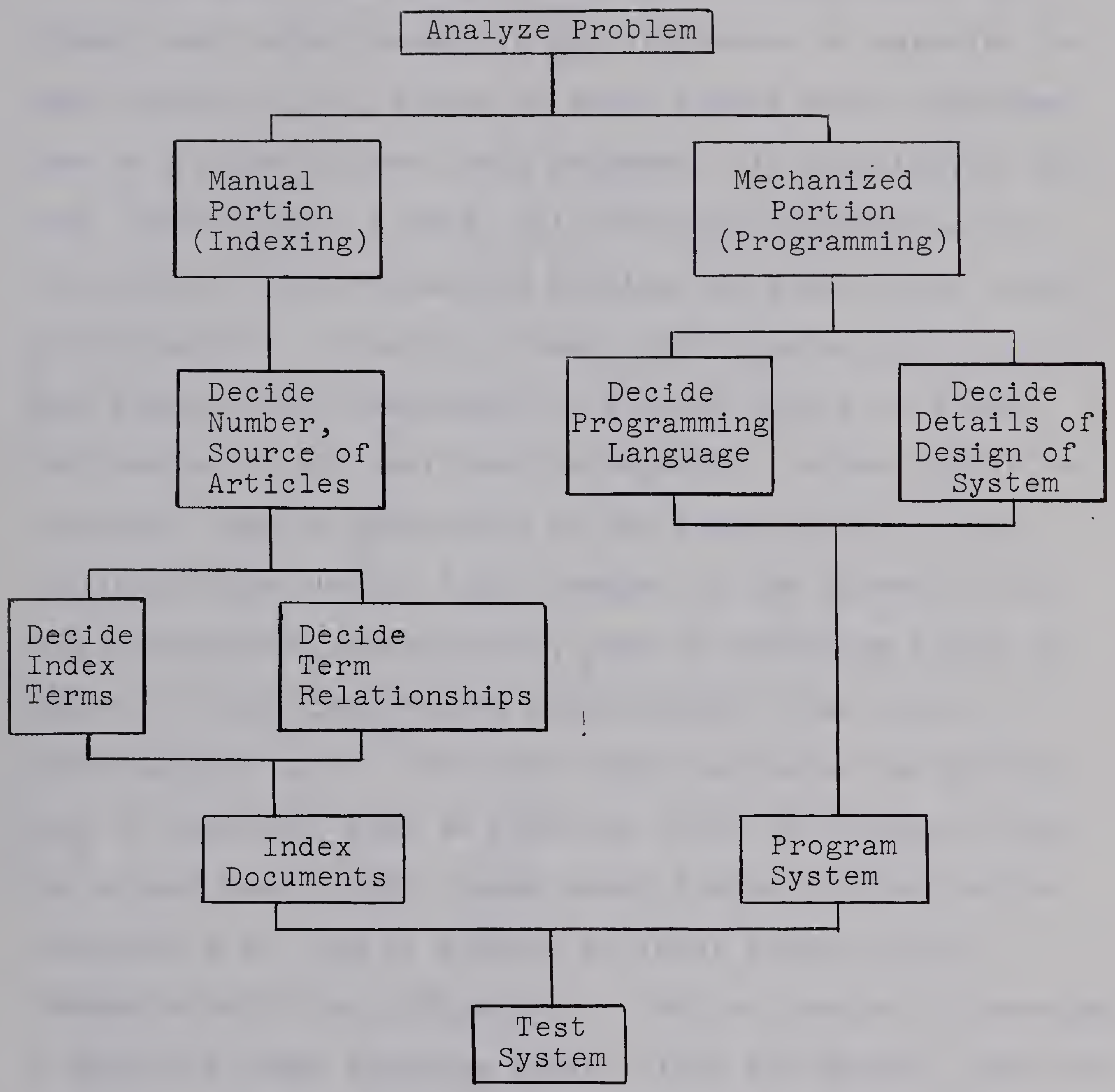
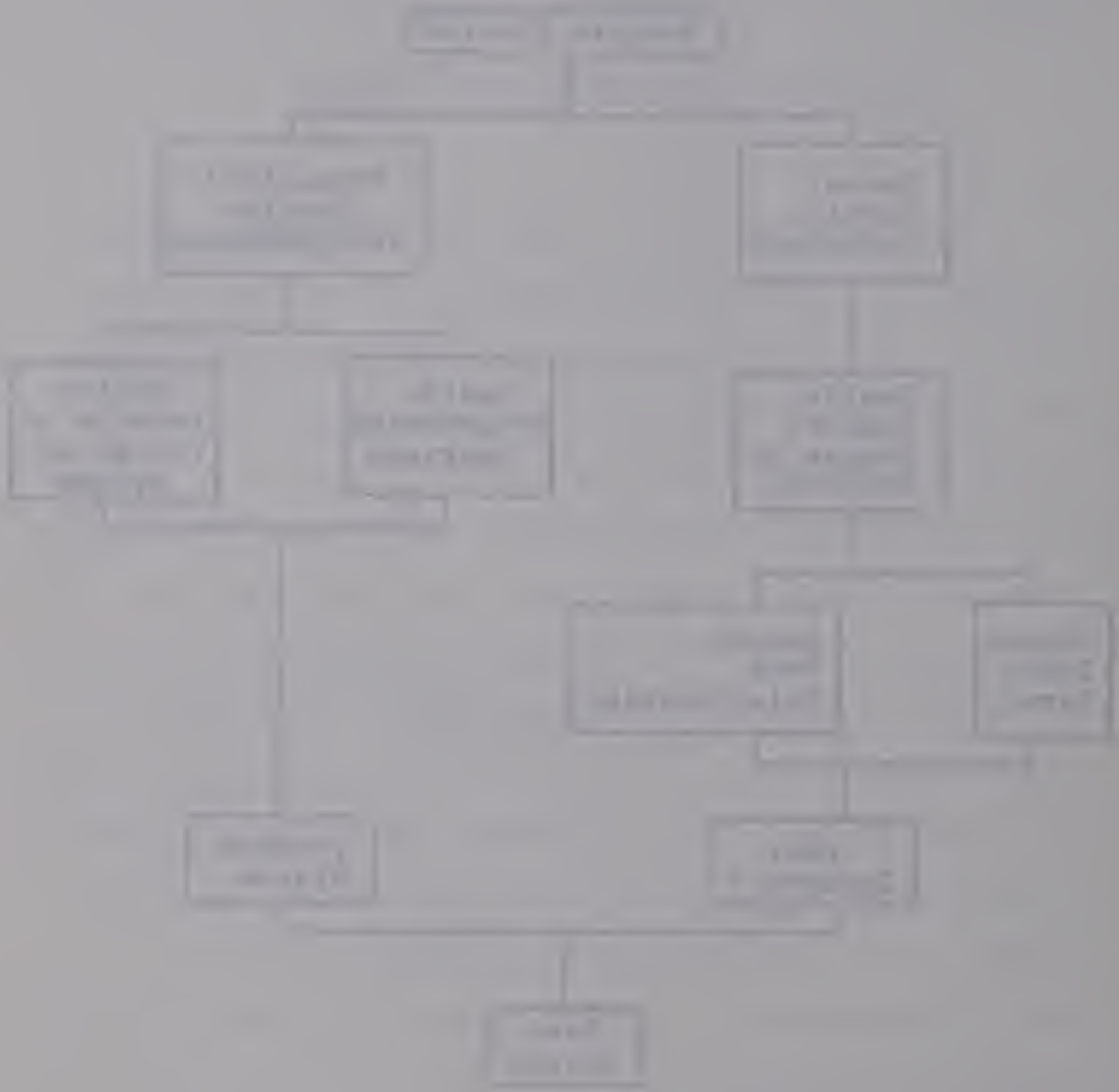

Analyze Problem
Manual Portion (Indexing)
Mechanized Portion (Programming)
Decide Number, Source of Articles
Decide Programming Language
Decide Details of Design of System
Decide Index Terms
Decide Term Relationships
Index Documents
Program System
Test System

Figure A.1

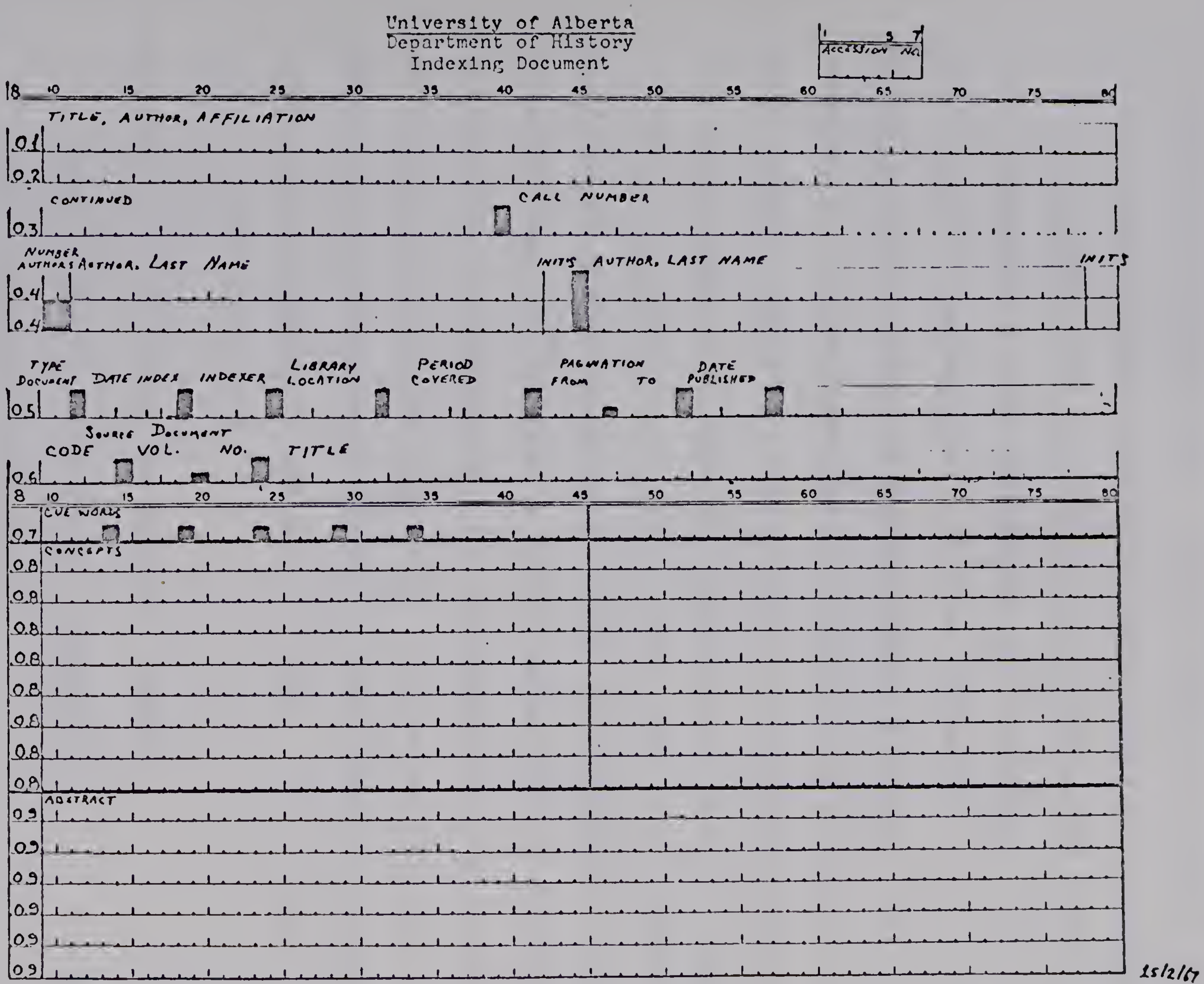
Statement and Solution of the Problem



[illegible]

[illegible]

The words "cue word", "key word", "descriptor" and "index term" often appear in the literature to describe the same concept, i.e., a word or short phrase which describes part or all the content of a document. In this thesis, the word "index term" is used. All decisions concerning the selection of index terms for testing the SARA system rested on one person. In order to check the indexing, opinions of more people with backgrounds in history should be sought. A publication of the American Bibliographic Center (1965) was consulted, and an adaptation of the index terms in this publication was used. Small changes in the format of the index terms were incorporated, such as replacing blanks by hyphens, to aid in computer manipulation. Two levels of indexing were used. The first level indicated the general area of interest, such as politics (POL) or religion (REL). The second level limits these broad fields to more narrow concepts, e.g., REL is limited by index terms such as "Roman-Catholic" and "Missions". The two levels of indexing, it was felt, made indexing easier since the general topic or topics of the article are required prior to those described by specific index terms. A coding sheet was designed by the author (see Figure A.2) onto which documents were indexed directly. Instructions for the correct use of this coding sheet must be provided for the indexer. Cards can be punched from this coding sheet, or the sheets may be used for direct input to the computer through the console. After choosing a sample of articles to be indexed, the history



University of Alberta
Department of History
Indexing Document

ACCESSION NO.

0.1 TITLE, AUTHOR, AFFILIATION
0.2

0.3 CONTINUED — CALL NUMBER

0.4 NUMBER AUTHORS — AUTHOR, LAST NAME — INITS — AUTHOR, LAST NAME — INITS
0.4

0.5 TYPE DOCUMENT — DATE INDEX — INDEXER — LIBRARY LOCATION — PERIOD COVERED — PAGINATION FROM — TO — DATE PUBLISHED

0.6 SOURCE DOCUMENT CODE — VOL. — NO. — TITLE

0.7 CUE WORDS

0.8 CONCEPTS
0.8
0.8
0.8
0.8
0.8
0.8
0.8

0.9 ABSTRACT
0.9
0.9
0.9
0.9
0.9

25/2/67

Figure A.2

Coding Sheet

participant chose index terms for the articles. If the need arose for a new index term, the term was added to the list of authorized index terms. After indexing several articles, the list was reviewed and synonymous words were combined. It was observed that the number of index terms added to the list decreased as more documents were indexed. About 100 articles were indexed; all but two appeared in the journals Agricultural History and Saskatchewan History. This sample averaged 6.7 index terms per document from a total of 250 index terms. The sample size used to test the system was relatively small. Of the 100 documents, 44 were selected to test the system. Not all of them could be used due to the lack of memory available to the SARA system. The test documents had 120 different index terms and 32 different authors from two journals; 59 of these terms were used in the manually generated hierarchy of term relations. All terms do not necessarily appear in the hierarchy since some terms are unrelated to the other terms. Following is a list of the terms used .

Cue words (referred to as general terms):

ALM (archives, libraries, museums)

BIO (biographic articles)

CUL (cultural life)

ECO (economic life)

EDU (education)

Cue words (continued)

FAM (family and genealogy)

GEO (geography)

IND (indians)

IRL (international relations and law)

LAN (land and agriculture)

LIT (literature)

MED (medicine and public health)

MET (methodology, research methods)

PER (period)

POL (politics, government)

POP (population, immigration)

REL (religions and churches)

SOC (social history, structure)

URB (urbanization, communities)

Concepts (referred to as specific terms):

ADMINISTRATION

AGHIST

AHENAKEW-CE

ALBERTA

ARCOLA

BALCARRES

BENNETT-TW

BETTER-FARMING-TRAINS

BOCKING-DH

Concepts (continued)

BROWN-GW

BRUNO

BUCK-RM

CATHOLIC-SETTLEMENT-SOCIETY

CHURCH-GC

CHURCH-OF-ENGLAND

CLERK-OF-LEGISLATIVE-ASSEMBLY

CLIMATE

CLOTHING

COLLEGE-OF-AGRICULTURE

COMARADES-OF-EQUITY

CONSTITUTION

CRIME

CULTIVATION

CUSTOMS

DAHLMAN-A

DAILY-PHOENIX

DAIRY

DIETARY

DIRECT-LEGISLATION-LEAGUE

DOCTOR

DOERFLER-B

DOMINION-LAND-SURVEY

EAGER-E

EAGLE-LAKE

Concepts (continued)

EASTEND

ELECTIONS

ENTERTAINMENT

EXPLORATION

EXPLORERS

FEDERAL-GOVERNMENT

FORT-LIVINGSTONE

FRENCH-CANADIAN

GRAVEL-LP

GRAVELBOURG

GRAYTOWN

GREENE-DL

GUERNSEY-GF

HAMILTON-ZM

HANSON-SD

HAULTAIN-FWG

IMMIGRATION

INDEPENDENT

INDIAN-DAY-SCHOOL

INDIAN-SCHOOL

INTERNATIONAL-BOUNDARY

JOHNSON-G

KIRK-LE

KLAUS-JF

KOESTER-CB

Concepts (continued)

LANG

LEGISLATIVE-ASSEMBLY

LIBRARIES

LITTLE-PINE

LOCAL-GOVERNMENT

MACDONALD-C

MACKAY-JA

MACLEAN-H

MANITOBAN

MATHESON-FAMILY

MILLER-AR

MISSIONARIES

MISSIONS

MORGAN-EC

MOTHERWELL-WR

MURRAY-JE

NATURAL-EVENTS

NISBET-J

NO-PARTY-LEAGUE

NORTH-QUAPPELLE

NORTHWEST-MOUNTED-POLICE

NORTHWEST-TERRITORIES

OLIVER-EH

ONION-LAKE

ORGANIZATIONS

Concepts (continued)

PALLISER-J

PARLIAMENTARY-SYSTEM

PARTIES

PATRONAGE

PEOPLES-POLITICAL-ASSOCIATION

PIONEER-LIFE

PLACE-NAMES

POLICE

POLICY

POLITICAL-THEORY

POLITICIANS

PRAIRIE-FIRE

PRESBYTERIAN

PRINCE-ALBERT

PUBLIC-UTILITIES

PUBLIC-WORKS

QUAPPELLE-VIDETTE

RECREATION

REGINA

REID-AN

ROE-FG

ROMAN-CATHOLIC

ROUMANIANS

SASKATCHEWAN

SASKATCHEWAN-HEARLD

Concepts (continued)

SASKATOON

SASKHIST

SCOT-W

SPAFFORD-DS

SPORTS

SPRY-IM

SPY-HILL-MUNICIPALITY

ST-STEPHEN-RM

STANLEY-MISSION

STEGNER-W

STEWART-EC

SURVEY

SUTHERLAND-W

SWAN-RIVER-BARRACKS

TERRITORIAL-GRAIN-GROWERS-ASSOCIATION

THE-LEADER

THOMPSON-WP

TRAVEL

TULLOCH-C

TURNER-AR

UNIVERSITIES

UNIVERSITY-OF-SASKATCHEWAN

URBAN

WEBSTER-EE

WILLOW-BRANCH

WOOD-MOUNTAIN

The documents used to test the system appeared in the two journals Saskatchewan History, volumes 9 to 19 inclusive and Agricultural History, volume 28.

The relationship between the terms was manually set up by the history participant. An adaptation of the hierarchy published by the American Bibliographic Center (1965) was used. Figure A.3 illustrates the terms used by the SARA system. This hierarchy is relatively simple due to the small sample size of documents and index terms used. It is characteristic of the social sciences and the humanities, such as history, that the hierarchy becomes much more complicated as the number of index terms increases. Hence, more research is required with a larger sample size on the approach taken in this thesis.

If an index term is to be added to the system, two steps must be taken. First, relationships between the term and all other index terms must be determined and the term must be manually entered in the hierarchy if applicable. Second, the term must be placed in the thesaurus and the hierarchial representation, *TREE*, within the computer. Presently, no routines exist which allow easy modification of the data within the computer. However, such routines could be coded. Addition of documents to the system is easily handled by using the *STORE* subsystem.

The selection and indexing portion of the SARA system is in the development stages; it is unpolished and, to a

large extent, untested. More research into this portion of the system would be beneficial. For example, extensive testing by members of the Department of History may indicate either areas of improvement in the choice of index terms and methods of indexing the documents, or a weakness in the hierarchial structure.

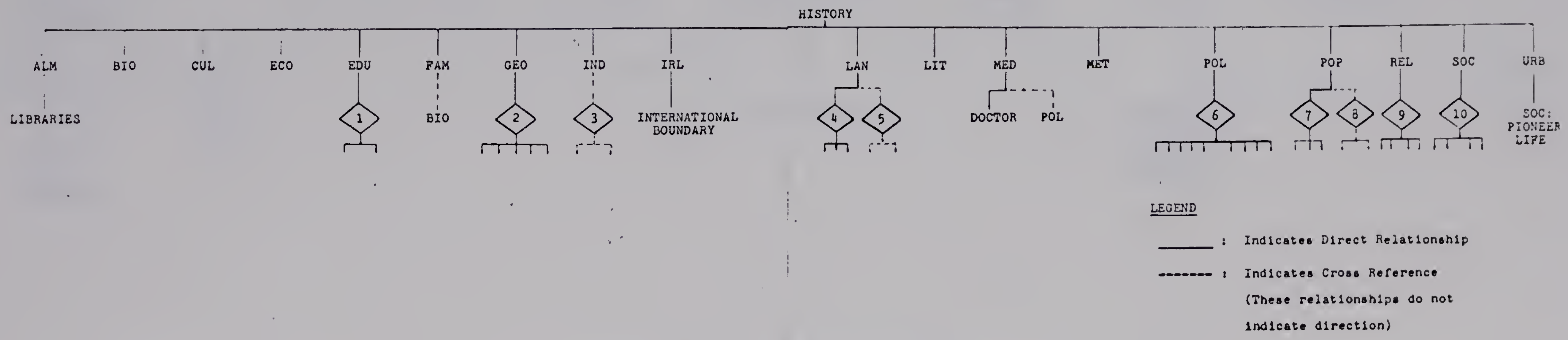


Figure A.3

Test Hierarchy

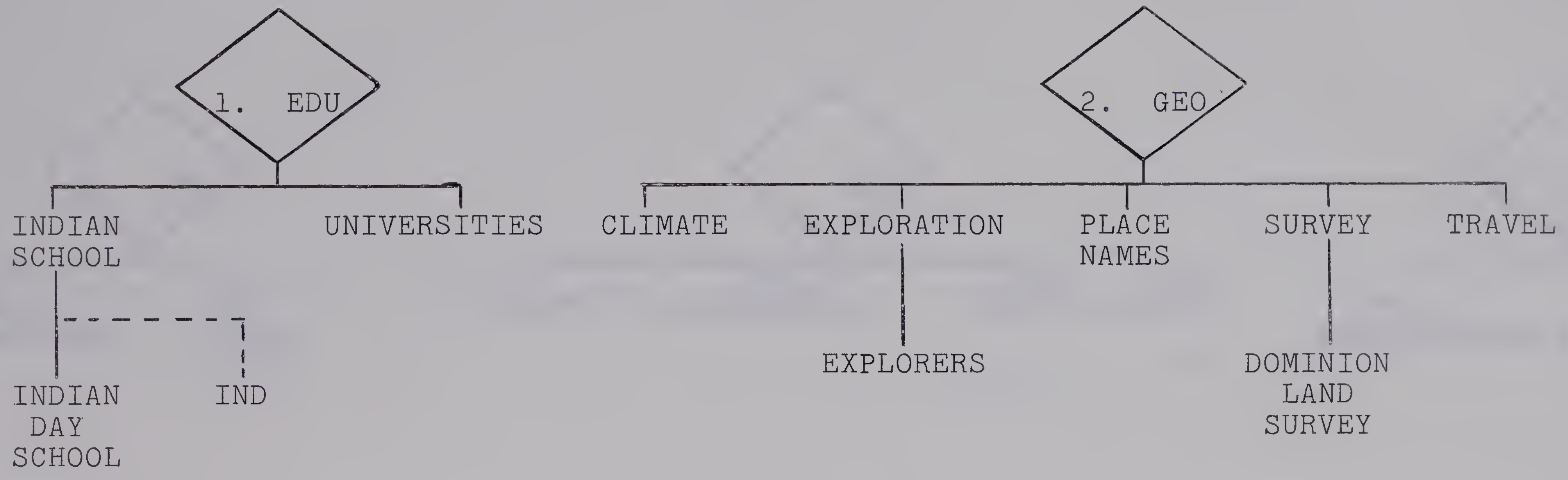


Figure A.3 (continued)

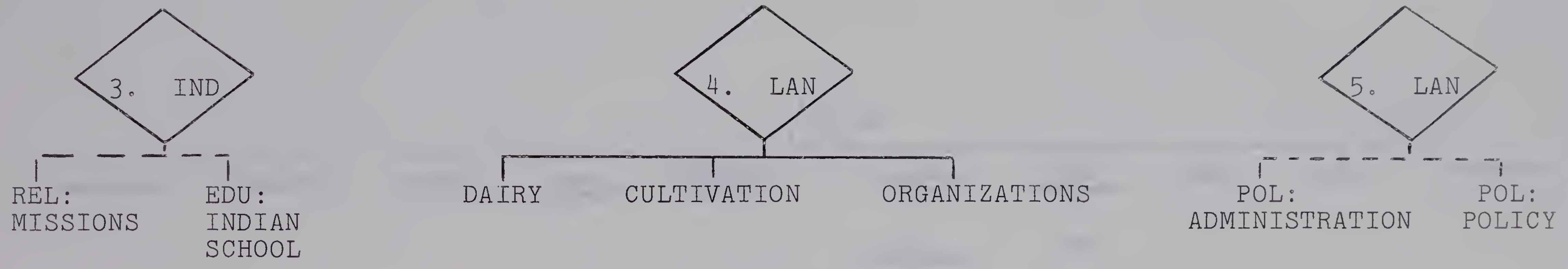


Figure A.3 (continued)

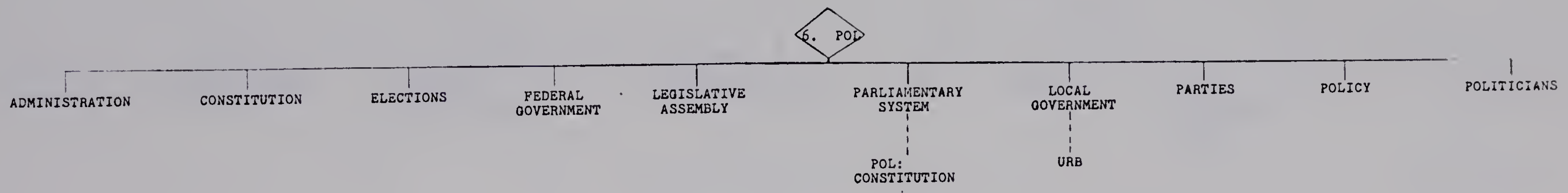


Figure A.3 (continued)

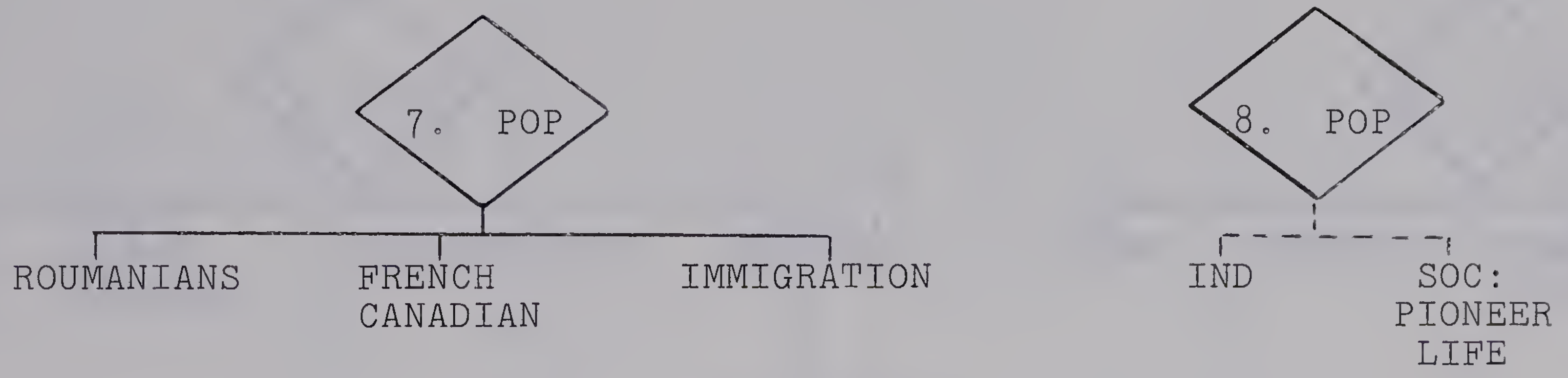


Figure A.3 (continued)

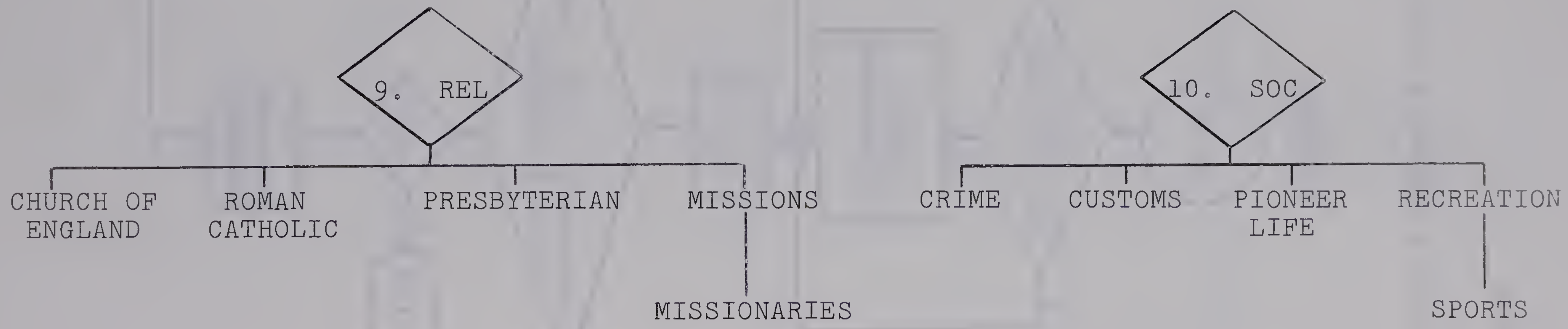


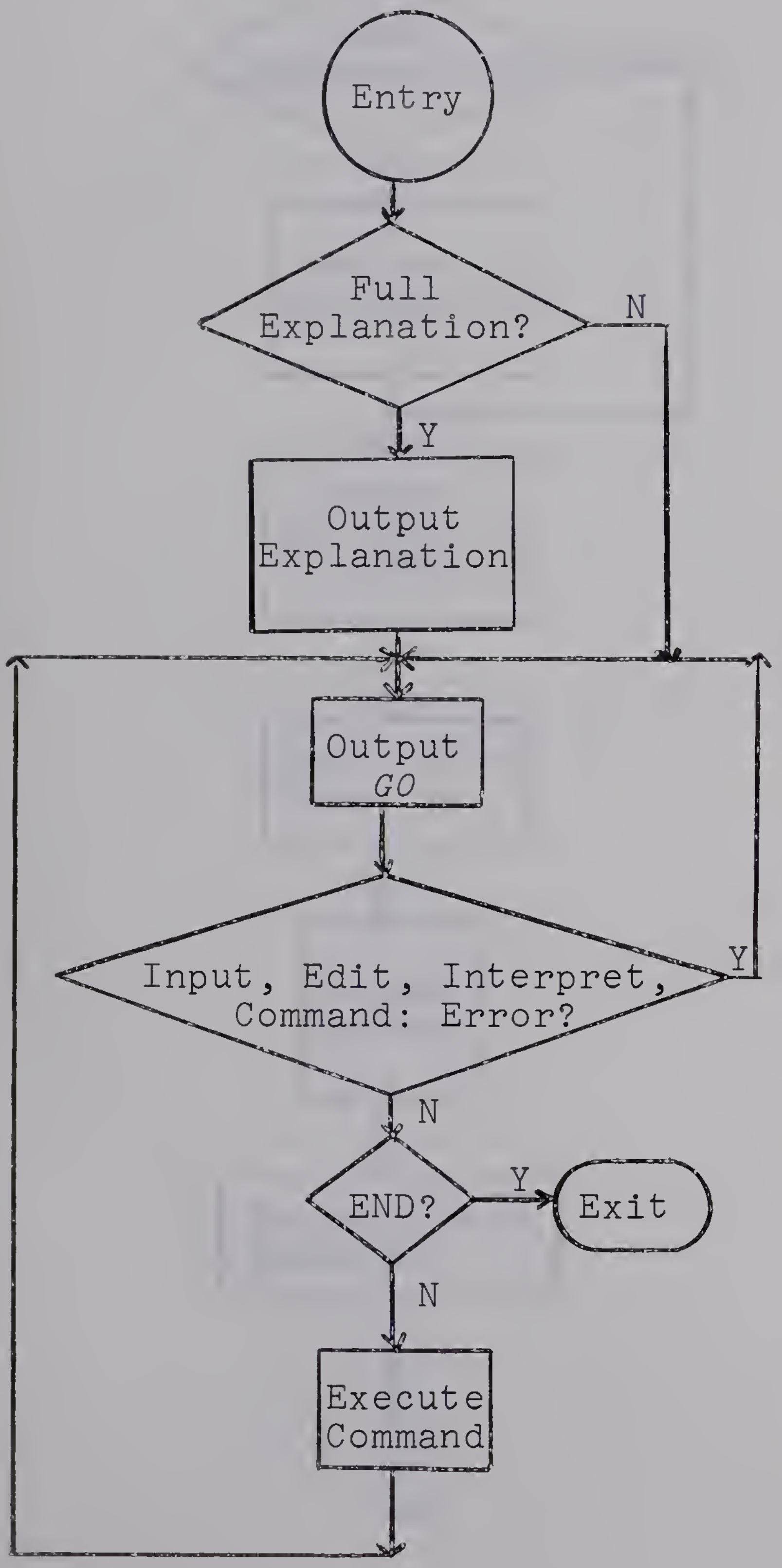
Figure A.3 (continued)

[illegible]

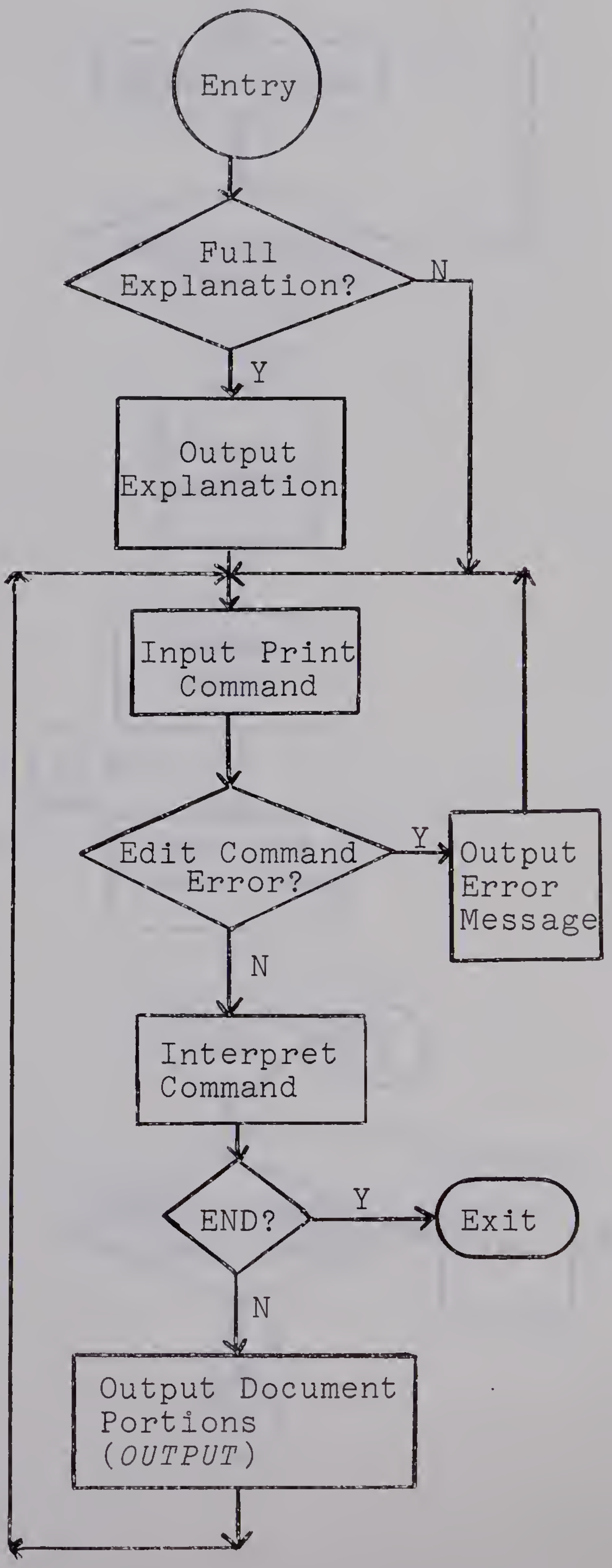
APPENDIX B

BLOCK DIAGRAMS AND LISTINGS OF ROUTINES IN SARA

SARACNL



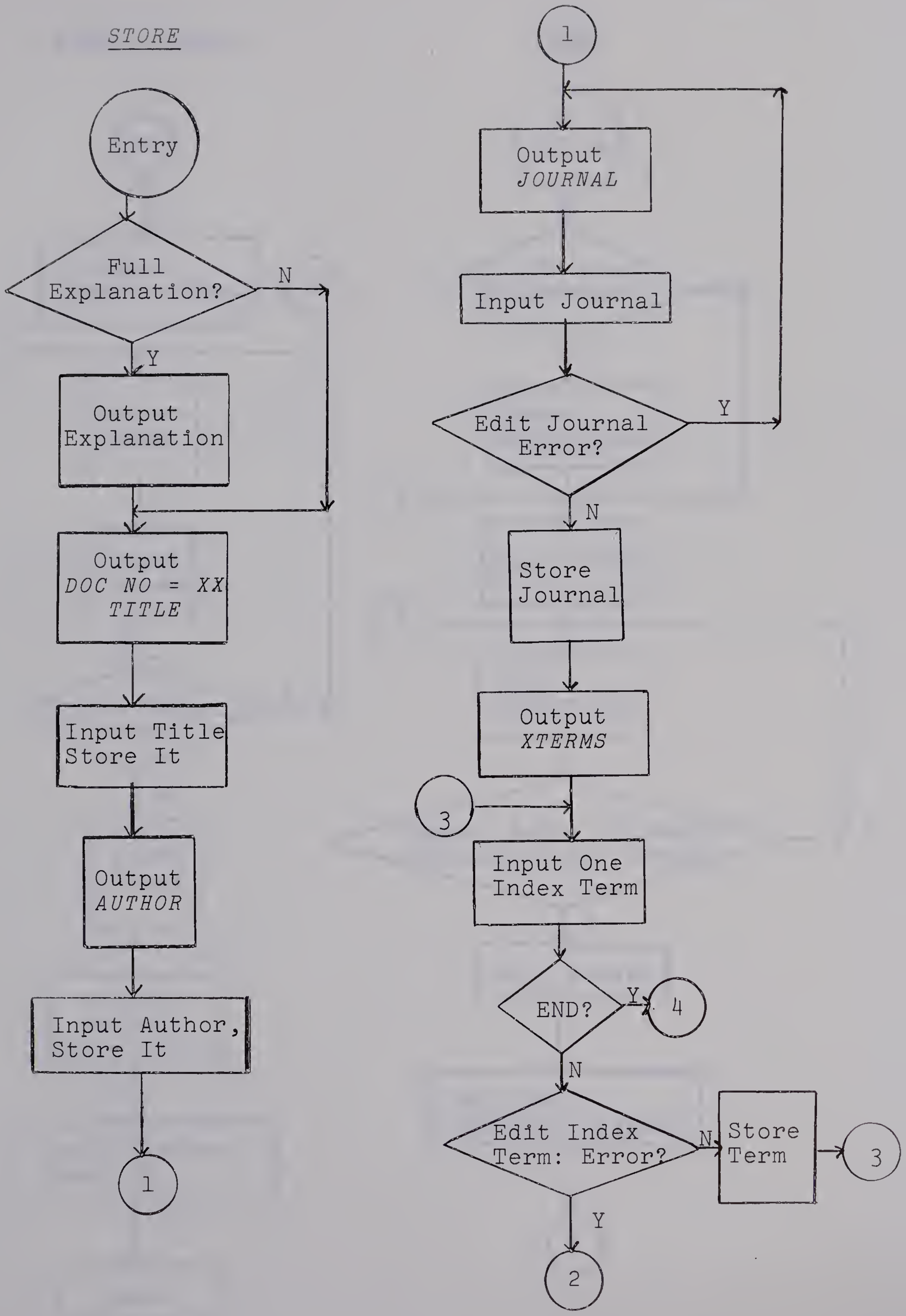
PRINT

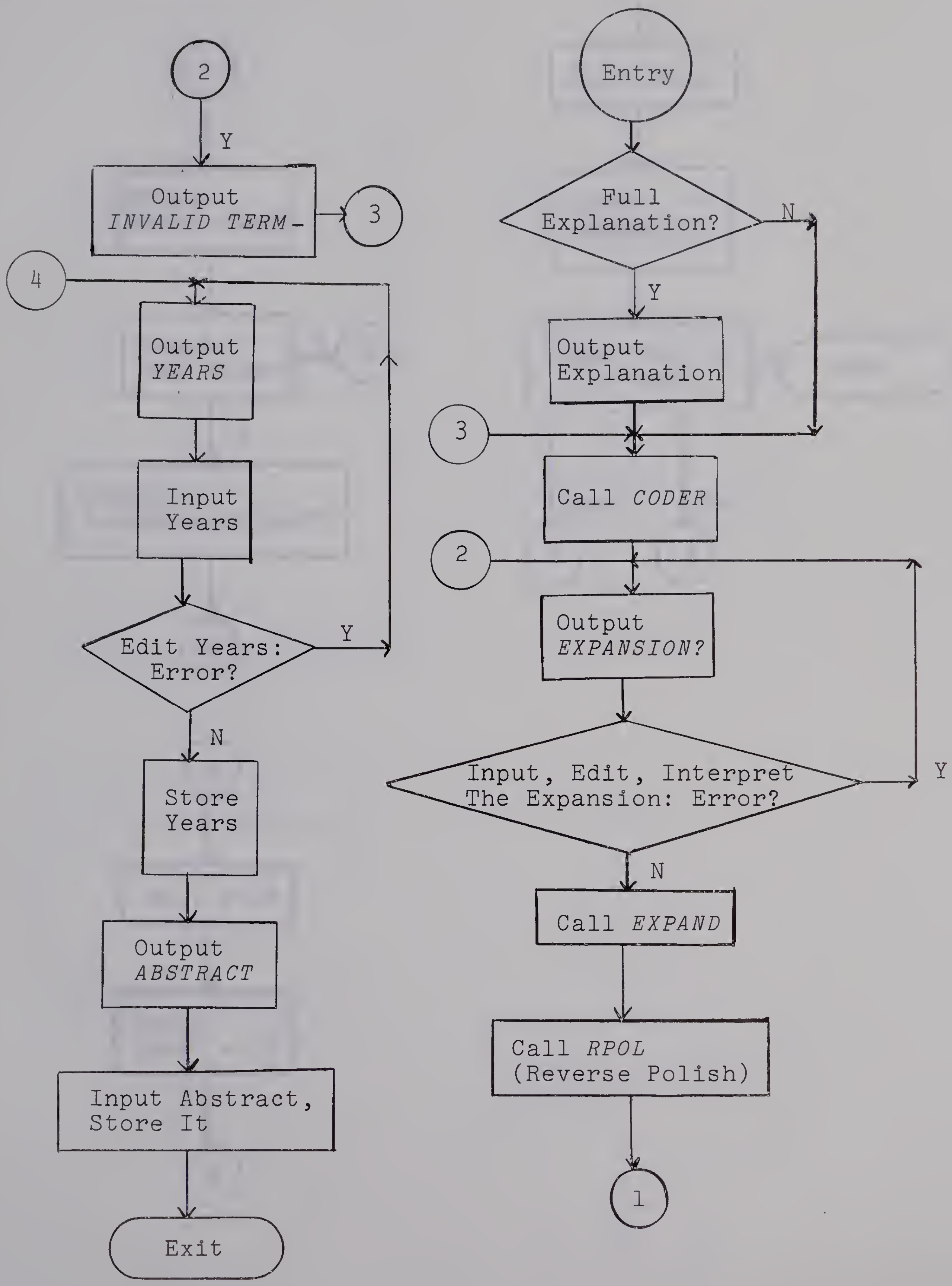


[illegible]

[illegible]

STORE

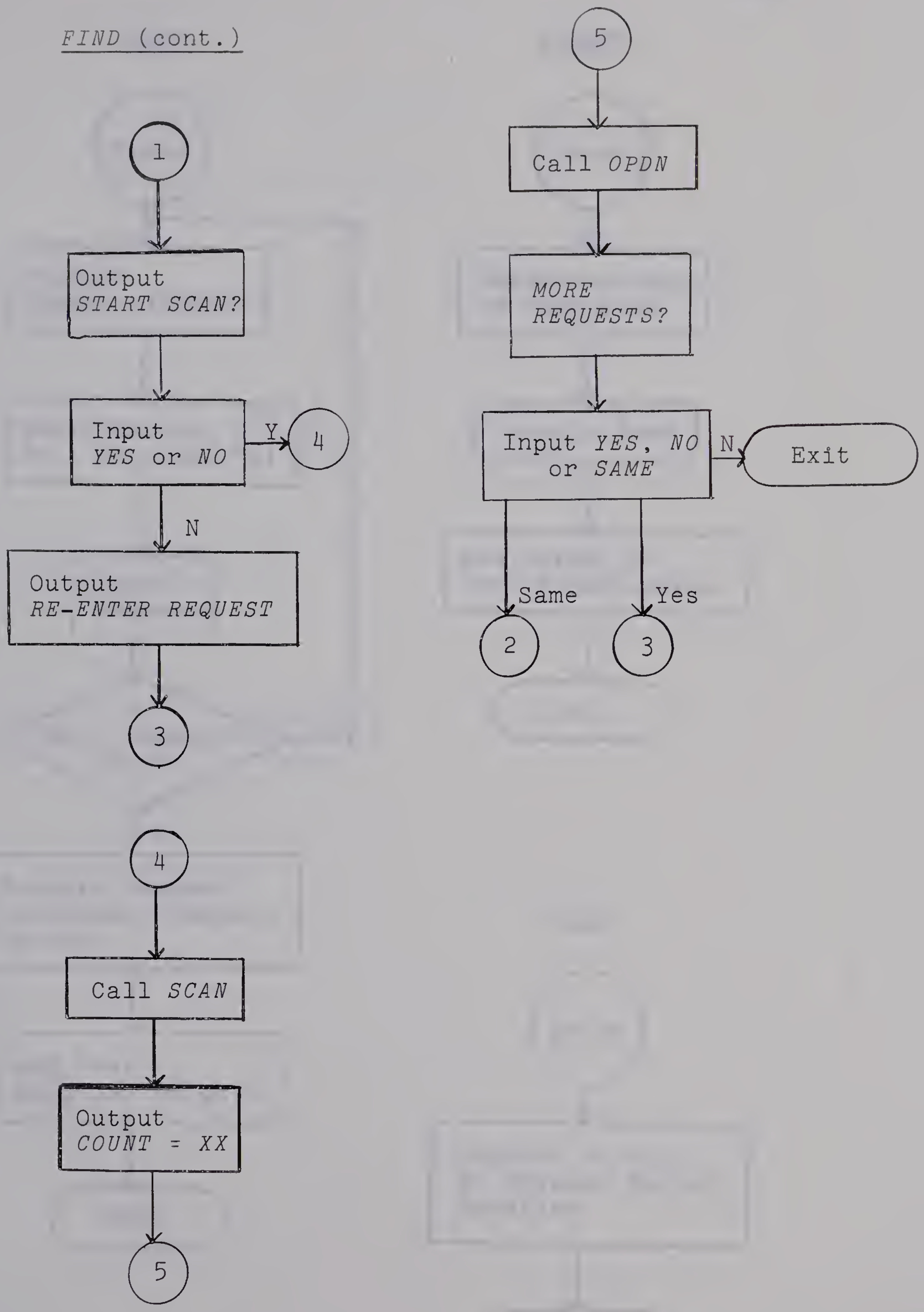

Entry
Full Explanation?
N
Y
Output Explanation
Output DOC NO = XX TITLE
Input Title Store It
Output AUTHOR
Input Author, Store It
1
1
Output JOURNAL
Input Journal
Edit Journal Error?
Y
N
Store Journal
Output XTERMS
3
Input One Index Term
END?
Y
4
N
Edit Index Term: Error?
N
Store Term
3
Y
2

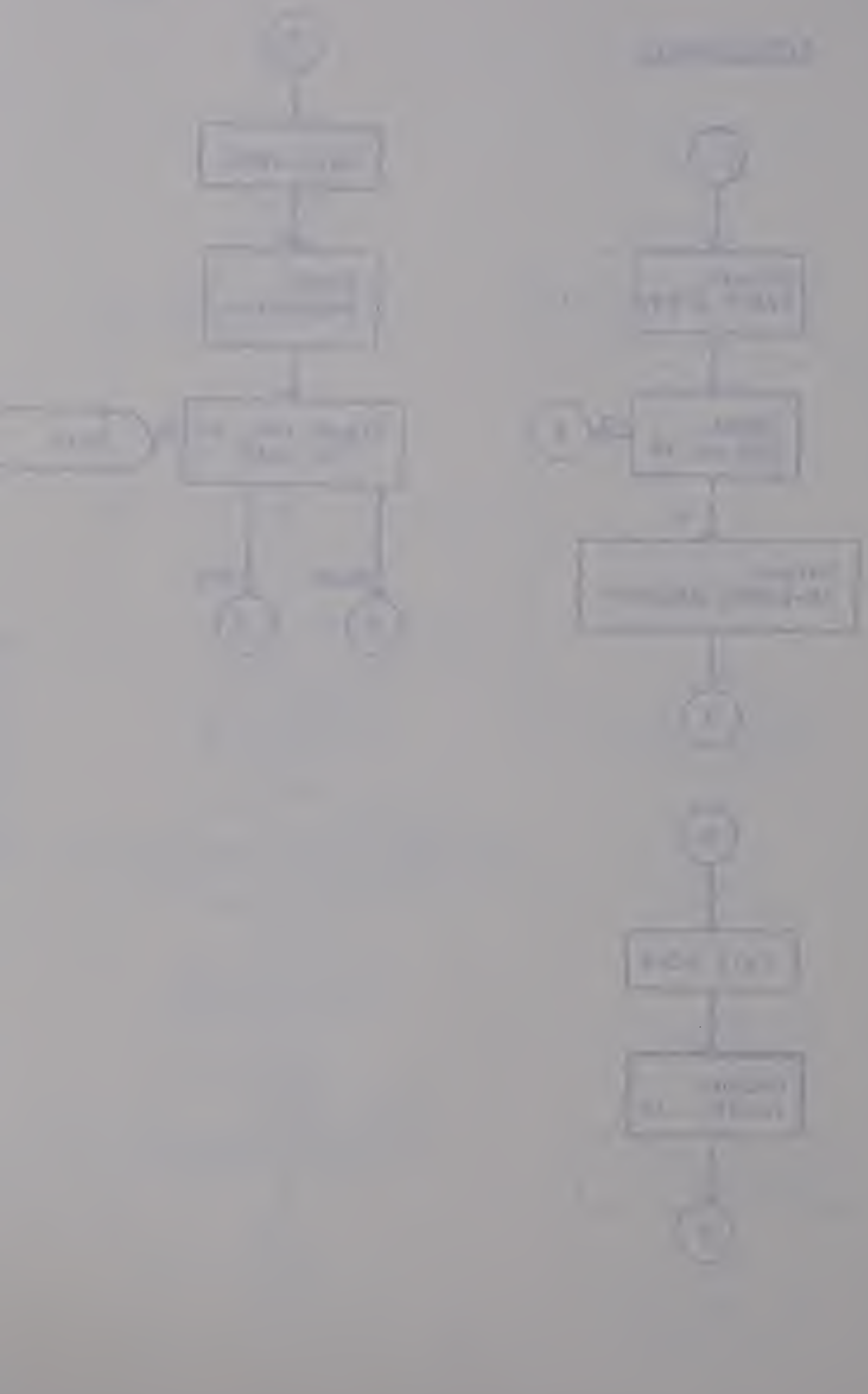


STORE (cont.)

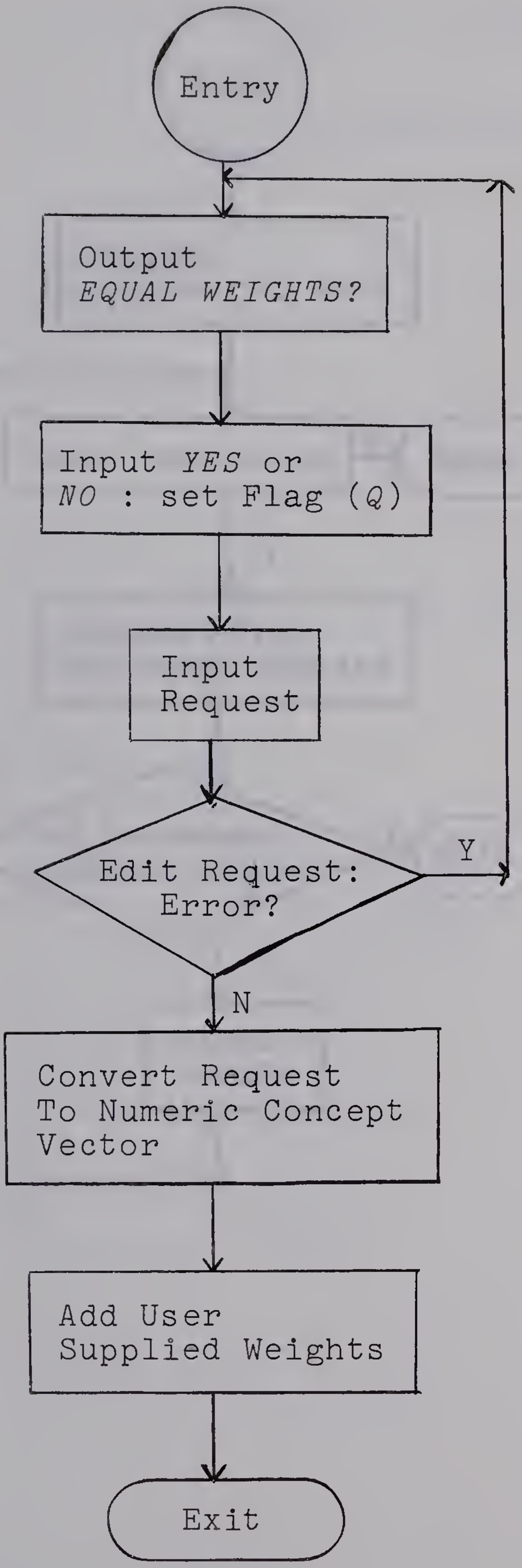
FIND

FIND (cont.)

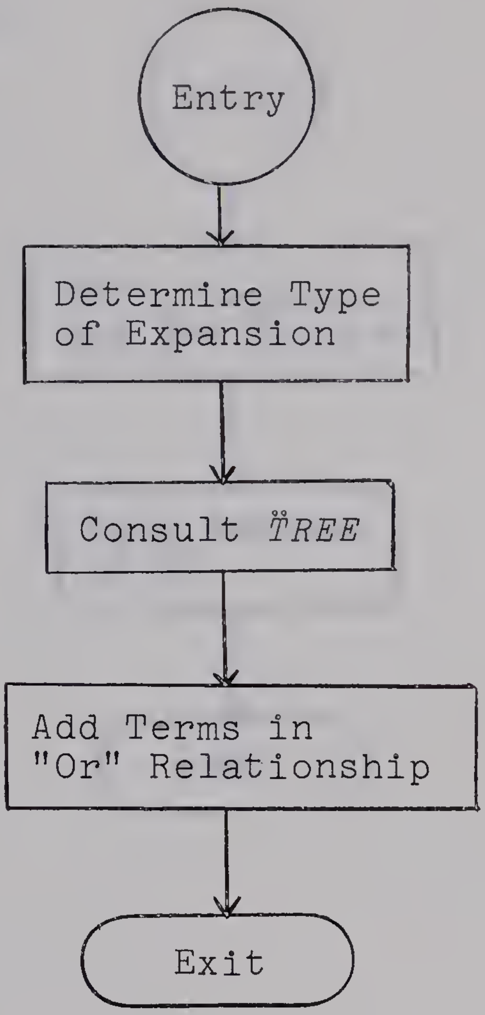

1
Output
START SCAN?
Input
YES or *NO*
Y
4
N
Output
RE-ENTER REQUEST
3
4
Call *SCAN*
Output
COUNT = XX
5
5
Call *OPDN*
MORE
REQUESTS?
Input *YES*, *NO*
or *SAME*
N
Exit
Same
Yes
2
3



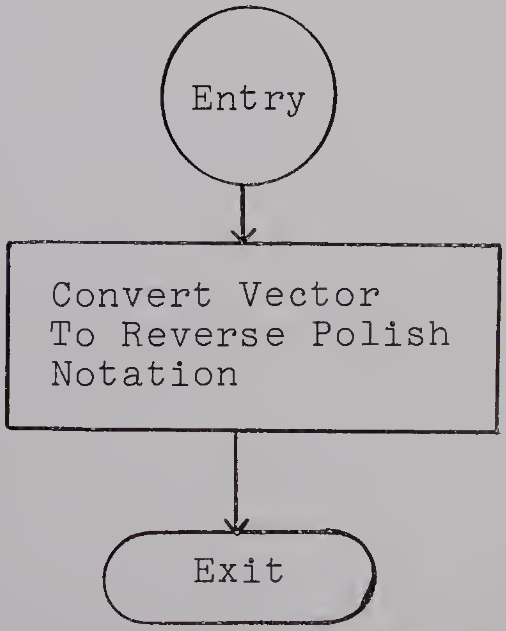
CODER



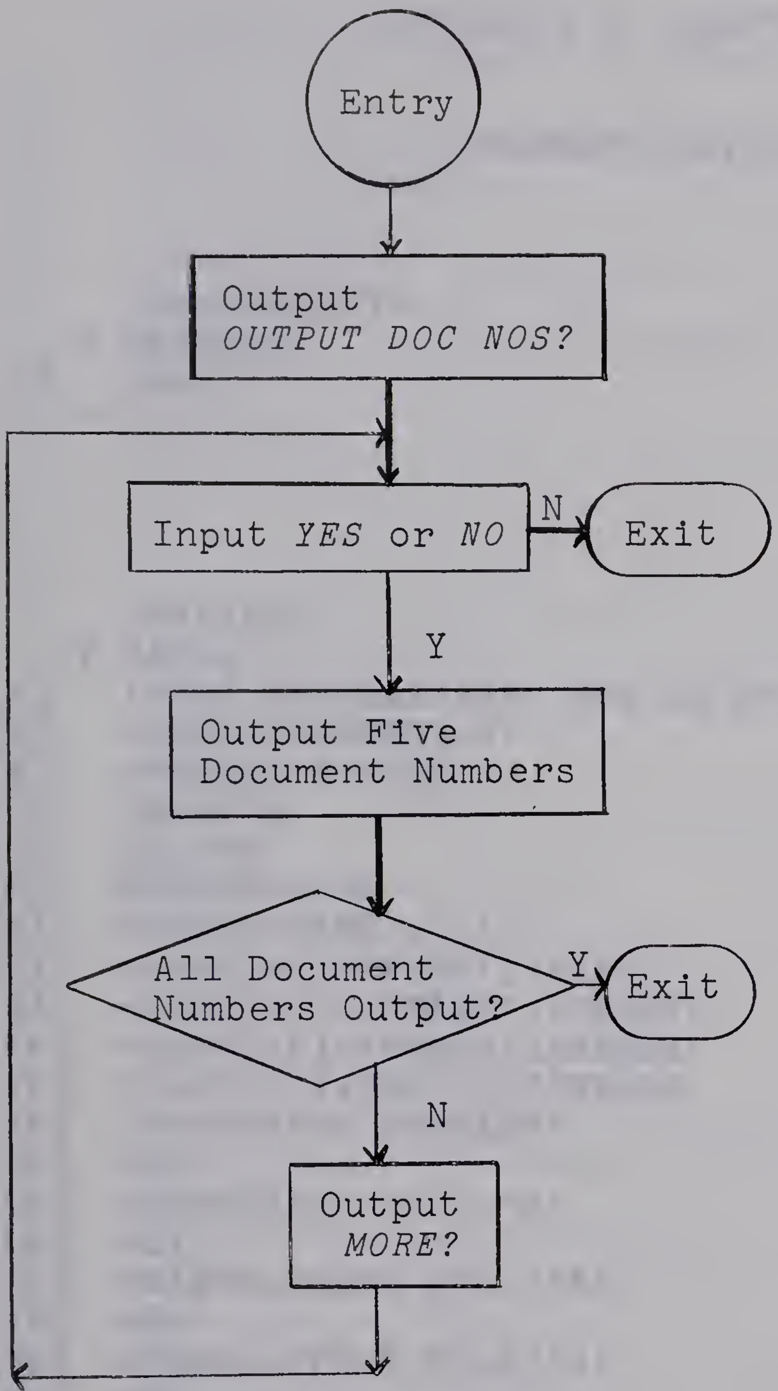
EXPAND



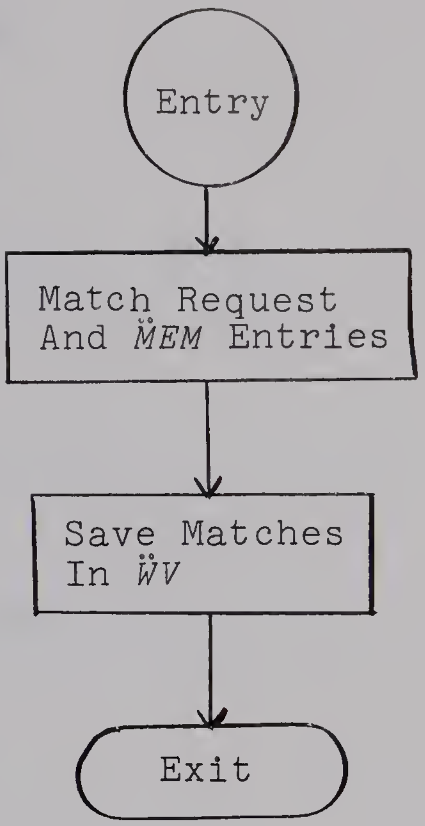
RPOL



OPDN


Entry
Output *OUTPUT DOC NOS?*
Input *YES* or *NO*
N
Exit
Y
Output Five Document Numbers
All Document Numbers Output?
Y
Exit
N
Output *MORE?*

SCAN


Entry
Match Request And *M̈EM* Entries
Save Matches In *ẄV*
Exit

APPENDIX B, CONTINUED

PROGRAM LISTINGS

```
       ∇SARACNL[⎕]∇
     ∇ SARACNL
[1]    SARA
     ∇
```

```
       ∇SARA[⎕]∇
     ∇ SARA;T
[1]    'FULL EXPLANATION? YES OR NO'
[2]    →(BEND T←INP⍞)/0
[3]    →(T[1]='N')/L1
[4]    XPLAN 1
[5]    L1:'GO'
[6]    S̈RANGE←ιρP̈T
[7]    T←7ρ(T←INP⍞),7ρ' '
[8]    →(∧/T[ι4]='FIND')/FINDL
[9]    →(∧/T[ι5]='PRINT')/PRINTL
[10]   →(∧/T[ι5]='STORE')/STOREL
[11]   →(∧/T[ι3]='END')/FINISHL
[12]   'INCORRECT COMMAND'
[13]   →L1
[14]   FINDL:FIND T[5]='N'
[15]   →L1
[16]   PRINTL:PRINT T[6]='N'
[17]   →L1
[18]   STOREL:STORE T[6]='N'
[19]   →L1
[20]   FINISHL:'ALL DONE'
     ∇
```

```
      ∇CODER[⎕]∇
    ∇ V←CODER X;T;WT;S;I;U;W;M;MP;P
[1]   T̈←V←ι0
[2]   CODER00:'EQUAL WEIGHTS?'
[3]   →(BEND Q←INP⍞)/0
[4]   Q←'N'=Q[1]
[5]   CODER13:T←'(',(T̈←⍞),')'
[6]   T←((T≠' ')/T),ι0
[7]   →((+/T='(')=+/T=')')/CODER14
[8]   'UNMATCHED PARENS'
[9]   →CODER13
[10]  CODER14:M←S\(S←∨/T∘.='(∨∧~<≤≠=≥>)')/T
[11]  MP←(~S)\(~S)/T
[12]  I←0
[13]  V←(~(¯1 SH S)∧S←M=' ')/M
[14]  V←(+/(V∘.=' (∨∧~<≤≠=≥>)')×((ρV),12)ρ-¯1+ι12),ι0
[15]  →(0≠ρV)/CODER99
[16]  V←0,ι0
[17]  CODER99:U←MP
[18]  CODER02:→(0=ρU)/0
[19]  →(0=ρU←(~(ρU)α¯1+(U≠' ')ι1)/U)/0
[20]  I←I+1
[21]  W←((ρU)α¯1+Uι' ')/U
[22]  U←(~(ρU)αUι' ')/U
[23]  →(~∧/(4ρW,' ')∊'0123456789')/CODER05
[24]  Y←NUM W
[25]  →(2100≤Y)/CODER15
[26]  →(0≠Y←Y÷10000)/CODER12
[27]  CODER05:→(¯1≠Y←AUTH W)/CODER07
[28]  CODER15:'ILLEGAL TERM - ',W
[29]  →CODER00
[30]  CODER07:→((~Q)∨∧/W∊'0123456789')/CODER021
[31]  CODER20:'WEIGHT? - ',W
[32]  WT←((WT≠' ')/WT←⍞),ι0
[33]  →(∧/11≠S←'0123456789'ιWT)/CODER08
[34]  '+VE NUMERALS ONLY'
[35]  →CODER20
[36]  CODER08:→(0.0001>WT←(NUM WT)×1E¯6)/CODER12
[37]  '≤99 ONLY'
[38]  →CODER20
[39]  CODER021:WT←1E¯6
[40]  CODER12:V[Vι0]←Y+WT
[41]  →CODER02
    ∇
```

```
      ∇EXPAND[⎕]∇
    ∇ X←P EXPAND Y;M;T;V;R
[1]    X←ι0
[2]    EXPAND03:→(0=ρY)/EXPAND05
[3]    →((1+ρY)=R←(Y>1)ι1)/EXPAND05
[4]    →(0=T←T̈PT[Y[R]-W←1|Y[R]])/EXPAND14
[5]    →((P=1),(P=2),P=3)/EXPAND11,EXPAND12,EXPAND21
[6]    EXPAND21:M←T̈REE[2;;T]
[7]    EXPAND13:→(∨/M)/EXPAND02
[8]    EXPAND14:X←X,Y[ιR]
[9]    →EXPAND06
[10]   EXPAND02:X←X,Y[ι¯1+R]
[11]   →(1=+/M)/EXPAND15
[12]   U←W+(∨/T̈PT∘.=M/ιρM)/ιρT̈PT
[13]   →EXPAND16
[14]   EXPAND15:U←W+T̈PTιM/ιρM
[15]   EXPAND16:V←((~T)×¯2)+(T←(2×ρU,ι0)ρ 0 1)\U
[16]   X←X,¯1,Y[R],V,¯11
[17]   EXPAND06:Y←(~(ρY)αR)/Y
[18]   →EXPAND03
[19]   EXPAND11:M←T̈REE[1;T;]
[20]   →EXPAND13
[21]   EXPAND12:M←T̈REE[1;;T]
[22]   →EXPAND13
[23]   EXPAND05:X←X,Y
    ∇
```

```
      ∇RPOL[⎕]∇
    ∇ S←RPOL I;PRI;OPS;T;TP
[1]    PRI← 0 0 1 1 2 1 1 1 1 1 1
[2]    OPS←-¯1+ι12
[3]    S←1ρ0
[4]    RPOL1:→(0=ρI)/RPOL6
[5]    T←I[1]
[6]    I←REST I
[7]    →(∧/T≠OPS)/RPOL5
[8]    TP←(T=OPS)/PRI
[9]    RPOL2:→(((T=¯4)∧S[1]=¯4)∨(T=¯1)∨TP>(S[1]=OPS)/PRI)/(
       RPOL3,RPOL4)[1+T≤¯11]
[10]   S←¯1 SH S
[11]   →RPOL2
[12]   RPOL3:S←T,S
[13]   →RPOL1
[14]   RPOL4:S←REST S
[15]   →RPOL1
[16]   RPOL5:S←S,T
[17]   →RPOL1
[18]   RPOL6:S←REST(-(((S=0)/ιρS)-1))SH S
    ∇
```

```
      ∇FIND[⎕]∇
    ∇ FIND X;T;W;V
[1]   →X/FIND01
[2]   'FIND FULL EXPLANATION? YES OR NO'
[3]   →(BEND T←INP⍞)/0
[4]   →(T[1]='N')/FIND01
[5]   XPLAN 2
[6]   FIND01:W←V←CODER X
[7]   →(0=ρV)/FIND11
[8]   FIND06:'EXPANSION?'
[9]   →(BEND T←(INP⍞)[⍳3])/0
[10]  T←((∧/'NON'=T),(∧/'SPE'=T),(∧/'GEN'=T),∧/'REL'=T)/
      0 1 2 3
[11]  →(0=ρT,⍳0)/FIND06
[12]  →(T=0)/FIND04
[13]  W←T EXPAND V
[14]  FIND04:W←RPOL W
[15]  'START SCAN?'
[16]  →(BEND T←INP⍞)/0
[17]  →(T[1]='Y')/FIND05
[18]  'RE-ENTER REQUEST'
[19]  →FIND01
[20]  FIND05:C←SCAN W
[21]  →(¯1≠C)/FIND03
[22]  'F SYNTAX ERROR'
[23]  T̈
[24]  →FIND01
[25]  FIND03:(('COUNT = ');C)
[26]  OPDN
[27]  FIND11:'MORE REQUESTS?'
[28]  →((T[1]='N')∨BEND T←INP⍞)/0
[29]  →(∧/'SAME'=T[⍳4])/FIND06
[30]  →FIND01
    ∇
```

```
      ∇SCAN[⎕]∇
    ∇ C←SCAN X;T;Y;Z
[1]   WVPT←1,⍳0
[2]   ẄV←⍳0
[3]   →(1<⍴X)/SCAN010
[4]   ẄV←(1≤T)/T←GETVEC X
[5]   →SCAN06
[6]   SCAN010:X←X,0
[7]   SCAN01:T←X[1]
[8]   →((T<0),T=0)/SCANLTZ,SCANEQZ
[9]   X←¯1 SH X
[10]  →SCAN01
[11]  SCANLTZ:→((11≤T)∨1≥T←|T)/SCANSYN
[12]  →(((T≠4)∧Z=¯1+⍴X,⍳0)∨(⍴X,⍳0)≤Z←X⍳0)/SCANSYN
[13]  →(T≤4)/SCAN05
[14]  →(∨/2<YP←X[(¯1+⍴X),⍴X])/SCANSYN
[15]  ẄV←ẄV,(GETYR X[1],YP)+0.0001×⌈/1|10000×YP
[16]  →SCAN08
[17]  SCAN05:Z←Z-ZP←1|Z←GETVEC X[⍴X]
[18]  →(T=4)/SCANOT
[19]  Y←Y-YP←1|Y←GETVEC X[¯1+⍴X]
[20]  YP[V]←YP[V←(V≤⍴Z)/⍳⍴Y]+ZP[(V≤⍴Z)/V←Z⍳Y]
[21]  →(SCANOR,SCANAND)[¯1+T]
[22]  SCANEQZ:X←REST X,⍳0
[23]  →(2≠⍴X,1)/SCANSYN
[24]  SCAN06:C←⍴ẄV,⍳0
[25]  →0
[26]  SCANOR:ẄV←ẄV,(Y+YP),(~∨/[1]Y∘.=Z)/Z+ZP
[27]  SCAN08:X←REST 1 SH X
[28]  ẄV←(ẄV≥1)/ẄV
[29]  SCAN09:X←(REST 1 SH X),-1+⍴WVPT
[30]  X←REST X
[31]  WVPT←WVPT,1+⍴ẄV,⍳0
[32]  →SCAN01
[33]  SCANAND:ẄV←ẄV,(∨/Y∘.=Z)/Y+YP
[34]  →SCAN08
[35]  SCANOT:→(0>⍴ZP,⍳0)/SCANOT01
[36]  ZP←0
[37]  SCANOT01:ẄV←ẄV,(⌈/ZP)+(~∨/Y∘.=Z)/Y←⍳M̈NDOC
[38]  →SCAN09
[39]  SCANSYN:C←¯1
    ∇
```

```
      ∇GETMEM[⎕]∇
    ∇ V←T GETMEM X;U;N
[1]   V←⍳0
[2]   GETMEM01:V←V,UρM̈EM[X;⍳U←T-1]
[3]   →(999999≠X←M̈EM[X;T])/GETMEM01
[4]   →(0=ρV)/0
[5]   →(∧/U←10000≤V)/0
[6]   V←((~U)×V)+U\(N-10000|N←U/V)÷10000
    ∇


      ∇GETVEC[⎕]∇
    ∇ V←GETVEC X;W
[1]   V←⍳0
[2]   →(0=ρX,⍳0)/0
[3]   X←X-W←1|X
[4]   →((X≥1)∧~X∊S̈RANGE)/0
[5]   →((X=0),X<0)/0,GETVEC01
[6]   X←P̈T[X]
[7]   V←W+N̈C GETMEM X
[8]   →0
[9]   GETVEC01:V←ẄV[W←¯1+WVPT[X-1]+⍳WVPT[X]-WVPT[(X←|X)-1]
      ]
[10]  WVPT←REST 1 SH WVPT
[11]  ẄV[W]←0
[12]  ẄV←(0≠ẄV)/ẄV
    ∇


      ∇GETYR[⎕]∇
    ∇ V←GETYR X;Y;Z
[1]   GETYR01←GETYR02←0
[2]   V←⍳0
[3]   →((Y>6)∨1>Y←¯4+|X[1])/0
[4]   →(1≠ρZ←⌊0.0001+10000×((1≠⌊X[2 3])/X[2 3]),⍳0)/0
[5]   →(SCANLT,SCANLE,SCANNE,SCANEQ,SCANGE,SCANGT)[Y]
[6]   SCANGT:V←V,(∨/(ŸMEM[; 2 3 4]≠999999)∧ŸMEM[;
      2 3 4]>Z)/ŸMEM[;1]
[7]   →GETYR01
[8]   SCANLT:V←V,(∨/((ŸMEM[; 2 3 4]≠0)∧ŸMEM[; 2 3
      4]≠1)∧ŸMEM[; 2 3 4]<Z)/ŸMEM[;1]
[9]   →GETYR02
[10]  SCANLE:GETYR02←GETYR12
[11]  →SCANLT
[12]  GETYR12:→SCANEQ
[13]  SCANGE:GETYR01←GETYR11
[14]  →SCANGT
[15]  GETYR11:→SCANEQ
[16]  SCANNE:V←V,(~EQF Z)/ŸMEM[;1]
[17]  →GETYR20
[18]  SCANEQ:V←V,(EQF Z)/ŸMEM[;1]
[19]  GETYR20:→(0=ρV←V,⍳0)/0
[20]  V←RMVCM V
    ∇
```

```
      ∇OPDN[⎕]∇
    ∇ OPDN;T;I
[1]   →(1=ρẄV,1)/0
[2]   'OUTPUT DOC NOS?'
[3]   →('N'=(INP⍞)[1])/0
[4]   ẄV←(ẄV-T)SORT T←⌊ẄV
[5]   →(∧/I←ẄV<10000)/OPDN3
[6]   ẄV←(I×ẄV)+(~I)×T-1|T←(I/ẄV)÷10000
[7]   OPDN3:I←0
[8]   OPDN1:→((ρẄV,⍳0)≤I←I+5)/OPDN2
[9]   (TR 2 5 ρT,ẄV[T←I+¯5+⍳5])
[10]  'MORE?'
[11]  →('N'=(INP⍞)[1])/0
[12]  →OPDN1
[13]  OPDN2:(TR(2,T)ρ(I+⍳T),ẄV[I+⍳T←(ρẄV,⍳0)-I←I-
      5])
    ∇
```

```
      ∇PRINT[⎕]∇
    ∇ PRINT X;T;U;V;AN;F;I
[1]   →X/PRINTL11
[2]   'FULL EXPLANATION? YES OR NO'
[3]   →(BEND T←INP⍞)/0
[4]   →(T[1]='N')/PRINTL11
[5]   XPLAN 3
[6]   PRINTL11:'
       PRINT WHAT?'
[7]   PRINT12:→(BEND U←INP⍞)/0
[8]   →(U[1]∊'0123456789')/PRINTL13
[9]   'DOC NO FIRST'
[10]  →PRINT12
[11]  PRINTL13:→(~∨/V←','=U)/PRINTL14
[12]  AN←NUM((ρU)α¯1+V⍳1)/U
[13]  F←5ρ0
[14]  I←+/V
[15]  PRINTL17:→(0>I←I-1)/PRINTL15
[16]  V←','=U
[17]  U←((~(ρU)αV⍳1)/U),3ρ' '
[18]  →(∧/U[⍳2]='AL')/PRINTL16
[19]  F←F∨(∧/V='TI'),(∧/V='AU'),(∧/V='JO'),(∧/V='XT'),(∧/'
      AB'=V←U[⍳2])
[20]  →PRINTL17
[21]  PRINTL16:F←5ρ1
[22]  PRINTL15:→(∨/F)/PRINTL18
[23]  PRINTL14:'INCORRECT PARAMETERS -- ',U
[24]  →PRINTL12
[25]  PRINTL18:AN OUTPUT F
[26]  →PRINTL11
    ∇
```

```
        ∇OUTPUT[⎕]∇
      ∇ AN OUTPUT X;M;TP
[1]     TP←(200α(M='|')⍳1)/M←(P̈ERMEM,199ρ'|')[P̈MPT[AN;
        2]+¯1+⍳200]
[2]     TP←(~TP='~')/TP
[3]     →(X[1]=0)/OUTPUT10
[4]     (((ρTP)α¯1+(TP='α')⍳1)/TP)
[5]     OUTPUT10:TP←(~(ρTP)α(TP='α')⍳1)/TP
[6]     →(X[2]=0)/OUTPUT11
[7]     (((ρTP)α¯1+(TP='∘')⍳1)/TP)
[8]     OUTPUT11:TP←(~(ρTP)α(TP='∘')⍳1)/TP
[9]     →(X[3]=0)/OUTPUT12
[10]    (((ρTP)α¯1+(TP='⊃')⍳1)/TP)
[11]    OUTPUT12:TP←(~(ρTP)α(TP='⊃')⍳1)/TP
[12]    →(X[4]=0)/OUTPUT13
[13]    →(~∨/M←AN=ŸMEM[;1])/OUTPUT14
[14]    (('YEARS - ');ŸMEM[M⍳1; 2 3 4])
[15]    OUTPUT14:M←'⊃',(((ρTP)α¯1+(TP='⊥')⍳1)/TP),'⊃'
[16]    M[(M='⊃')/⍳ρM]←' '
[17]    M
[18]    OUTPUT13:TP←(~(ρTP)α(TP='⊥')⍳1)/TP
[19]    →(X[5]=0)/0
[20]    (((ρTP)α¯1+(TP='|')⍳1)/TP)
      ∇
```

Page 144
listing of OUTPUT

```
      ∇STORE[⎕]∇
    ∇ STORE P;T
[1]   →P/STORE01
[2]   'FULL EXPLANATION?'
[3]   →(BEND T←INP⍞)/0
[4]   →(T[1]='N')/STORE01
[5]   XPLAN 4
[6]   STORE01:(('DOC NO ');M̈NDOC←M̈NDOC+1)
[7]   STRTIT
[8]   STRAUT
[9]   STRJOU
[10]  STRXTR
[11]  STRYR
[12]  STRABS
[13]  CLENUP
[14]  'MORE?'
[15]  →('Y'=(INP⍞)[1])/STORE01
    ∇


      ∇STRTIT[⎕]∇
    ∇ C←STRTIT;T
[1]   C←⍳0
[2]   'TITLE'
[3]   T←⍞,⍳0
[4]   →(0=⍴T)/STRTIT01
[5]   P̈ERMEM←P̈ERMEM,'~',T
[6]   →0
[7]   STRTIT01:P̈ERMEM←P̈ERMEM,'~NONE'
    ∇


      ∇STRMEM[⎕]∇
    ∇ C STRMEM T;CT
[1]   C←P̈T[C]
[2]   STRMEM01:→(999999=CT←M̈EM[C;N̈C])/STRMEM02
[3]   →(C=C←CT)/STRMEM01
[4]   STRMEM02:→(N̈C≤CT←M̈EM[C;⍳N̈C-1]⍳0)/STRMEM03
[5]   →(T=M̈EM[C;CT]←T)/0
[6]   STRMEM03:M̈EM←(((⍴M̈EM)[1]+1),N̈C)⍴(,M̈EM),T,((N̈C-
      2)⍴0),999999
[7]   M̈EM[C;N̈C]←(⍴M̈EM)[1]
    ∇
```

```
      ∇STRAUT[⎕]∇
    ∇ Z←STRAUT;T;C;U
[1]   Z←⍳0
[2]   'AUTHOR'
[3]   U←T←((T≠' ')/T←⍞),⍳0
[4]   →(0=⍴T)/STRAUT03
[5]   →(~',',∊T)/STRAUT05
[6]   T←T[⍳¯1+T⍳',']
[7]   STRAUT05:→(¯1=C←AUTH T)/STRAUT01
[8]   STRAUT02:C STRMEM M̈NDOC
[9]   →STRAUT04
[10]  STRAUT03:U←T←'NONE'
[11]  →STRAUT04
[12]  STRAUT01:ÄPT←ÄPT,1
[13]  M̈EM←(((⍴M̈EM)[1]+1),N̈C)⍴(,M̈EM),M̈NDOC,((N̈C-2)⍴0),
      999999
[14]  L̈ST←L̈ST,T
[15]  L̈STPT←L̈STPT,L̈STPT[⍴L̈STPT]+⍴T
[16]  P̈T←P̈T,(⍴M̈EM)[1]
[17]  STRAUT04:P̈ERMEM←P̈ERMEM,'α',U
    ∇



      ∇STRJOU[⎕]∇
    ∇ Z←STRJOU;C;V
[1]   Z←⍳0
[2]   'JOURNAL'
[3]   STRJOU04:T←((V≠' ')/V←⍞),⍳0
[4]   →(0=⍴T)/STRJOU01
[5]   U←0
[6]   →(~','∊T)/STRJOU05
[7]   U←NUM(T∊'0123456789')/T
[8]   →(9999≥U)/STORE11
[9]   'VOLUME NO ≤ 9999'
[10]  →STRJOU04
[11]  STORE11:T←T[⍳¯1+T⍳',']
[12]  STRJOU05:→(¯1=C←AUTH T)/STRJOU02
[13]  C STRMEM U+M̈NDOC×10000
[14]  →0
[15]  STRJOU01:V←'NONE'
[16]  →STRJOU03
[17]  STRJOU02:'INVALID JOURNAL - ',V
[18]  →STRJOU04
    ∇
```

```
       ∇STRXTR[⎕]∇
     ∇ X←STRXTR;T;X
[1]    Z←ι0
[2]    'XTERMS'
[3]    STRXTR02:T←((T≠' ')/T←⍞),ι0
[4]    →(0=ρT)/STRXTR02
[5]    →(BEND(T,'   ')[ι3])/0
[6]    →(¯1∊C←AUTH T)/STRXTR01
[7]    C STRMEM M̈NDOC
[8]    P̈ERMEM←P̈ERMEM,'⊃',T
[9]    →STRXTR02
[10]   STRXTR01:'INVLAID XTERM - ',T
[11]   →STRXTR02
     ∇


       ∇STRYR[⎕]∇
     ∇ C←STRYR;T;I
[1]    C←ι0
[2]    'YEARS'
[3]    T←((T≠' ')/T←⍞),ι0
[4]    →(~(T,' ')[1]∊'0123456789')/0
[5]    →(0=ρT)/0
[6]    ŸMEM←(((ρŸMEM)[1]+1),4)ρ(,ŸMEM),M̈NDOC,3ρ999999
[7]    →('-'∊T)/STRYR01
[8]    I←1
[9]    STRYR03:→(4<I←I+1)/0
[10]   →(0=ρT)/0
[11]   ŸMEM[(ρŸMEM)[1];I]←NUM T[ι4]
[12]   T←(~(ρT)α5)/T
[13]   →STRYR03
[14]   STRYR01:ŸMEM[(ρŸMEM)[1]; 2 3 4]←(NUM T[ι4]),0,NUM T[
       5+ι4]
     ∇
```

```
      ∇STRABS[⎕]∇
    ∇ C←STRABS;T
[1]   C←⍳0
[2]   'ABSTRACT'
[3]   T←((T≠' ')/T←⍞),⍳0
[4]   →(0>⍴T)/STRABS01
[5]   T←'NONE'
[6]   STRABS01:P̈ERMEM←P̈ERMEM,'⊥',T,'|'
    ∇

      ∇CLENUP[⎕]∇
    ∇ C←CLENUP
[1]   C←⍳0
[2]   P̈MPT←P̈MPT,1+⍴P̈ERMEM
    ∇
```

```
      ∇AUTH[⎕]∇
    ∇ V←AUTH W;RW;AV;N
[1]   RW←ρ(W←(W≠' ')/W),⍳0
[2]   AV←L̈STPT[1+N]-L̈STPT[N←⍳¯1+ρL̈STPT]
[3]   AV←(RW=AV)/⍳ρAV
[4]   →(0=ρAV)/AUTHAUTH01
[5]   I←0
[6]   AUTHAUTH02:→((ρAV,⍳0)<I←I+1)/AUTHAUTH01
[7]   →(~∧/W=L̈ST[¯1+L̈STPT[(AV,⍳0)[I]]+⍳RW])/AUTHAUTH02
[8]   V←AV[I]
[9]   →0
[10]  AUTHAUTH01:V←¯1
    ∇
```

```
      ∇BEND[⎕]∇
    ∇ X←BEND T
[1]   X←∧/'END'=(T,'   ')[⍳3]
    ∇
```

```
      ∇EQF[⎕]∇
    ∇ T←EQF Z
[1]   T←∨/Z=ŸMEM[; 2 3 4]
[2]   T←T∨(ŸMEM[;3]=0)∧(ŸMEM[;4]>Z)∧(ŸMEM[;4]≠999999)∧ŸMEM
      [;2]<Z
    ∇
```

```
      ∇INP[⎕]∇
    ∇ T←INP X
[1]   T←((X≠' ')/X),'     '
    ∇
```

```
      ∇NUM[⎕]∇
    ∇ V←NUM M
[1]   V←10⊥¯1+'0123456789'⍳M
    ∇
```

```
      ∇REST[⎕]∇
    ∇ X←REST Y
[1]   X←(~(ρY)α1)/Y
    ∇
```

```
      ∇RMVCM[⎕]∇
    ∇ N←RMVCM V
[1]   N←(N∊V,ι0)/N←ι⌈/V,ι0
    ∇
```

```
      ∇SH[⎕]∇
    ∇ SHZ←N SH V
[1]   SHZ←(V,ι0)[1+(ρ(V,ι0))|(ιρ(V,ι0))-1+N]
    ∇
```

```
      ∇TR[⎕]∇
    ∇ Z←TR M
[1]   Z←(ρM)[2 1]ρ(,M)[,(ι(ρM)[2])∘.+(ρM)[2]×¯1+ι(ρM)[1]]
    ∇
```

```
      ∇SORT[⎕]∇
    ∇ R←KEY SORT ITEM
[1]   R←ITEM[(+/(KEY∘.<KEY)+(KEY∘.=KEY)∧R∘.≤R)ιR←ιρKEY]
    ∇
```

```
      ∇XPLAN[⎕]∇
    ∇ V←XPLAN P
[1]   V←⍳0
[2]   →(XPLAN1,XPLAN2,XPLAN3,XPLAN4)[P]
[3]   XPLAN1:'
      YOU ARE UNDER THE CONTROL OF THE MAINLINE ROUTINE'
[4]   'YOU HAVE 3 ROUTINES AVAILABLE FOR PROCESSING'
[5]   'IN ORDER TO ENTER THE REQUIRED ROUTINE, TYPE ITS NA
      ME AFTER'
[6]   '      ''GO'' IS TYPED'
[7]   'IN ORDER TO RETURN CONTROL TO THIS ROUTINE, TYPE ''
      END'' AT'
[8]   '      ANY INPUT TIME
      '
[9]   'FIND'
[10]  '      ACCEPTS INPUT REQUESTS AND RETURNS DOCUMENT NU
      MBERS'
[11]  '      SATISFYING THE REQUEST'
[12]  'PRINT'
[13]  '      LISTS ANY PART OF THE DOCUMENT WITH THE GIVEN
      DOCUMENT'
[14]  '      NUMBER'
[15]  'STORE'
[16]  '      ALLOWS STORAGE OF DOCUMENTS'
[17]  'END'
[18]  '      RETURNS CONTROL TO THE APL SYSTEM'
[19]  →0
[20]  XPLAN2:'
      IF WEIGHTS ARE TO BE ATTACHED TO EACH INDEX TERM, TY
      PE ''YES'''
[21]  '      WHEN REQUESTED (EQUAL WEIGHTS?);'
[22]  '      ELSE,''NO'' '
[23]  'THEN, TYPE THE REQUEST IN STANDARD BOOLEAN FORM'
[24]  'WHEN ''EXPANSION?'' IS REQUESTED, ONE OF FOUR WORDS
       (OR THEIR'
[25]  '      ABBREVIATIONS AS IN THE FOLLOWING PARENTHESES)'
[26]  '      IS EXPECTED'
[27]  '
      NONE (NON)'
[28]  '      NO EXPANSION OF THE REQUEST'
[29]  'GENERAL (GEN)'
[30]  '      INCLUDE GENERAL TERMS IN AN OR RELATIONSHIP'
[31]  'SPECIFIC (SPE)'
[32]  '      INCLUDE SPECIFIC TERMS IN AN OR RELATIONSHIP'
[33]  'RELATED (REL)'
[34]  '      INCLUDE RELATED TERMS IN AN OR RELATIONSHIP
      '
[35]  'WHEN REQUESTED ''START SCAN?'' TYPE ''YES'' OR ''NO'
```

```
      '. IF'
[36]  '     ''YES'', A SEARCH OF ALL DOCUMENTS IS STARTED;
       IF ''NO'','
[37]  '     A REQUEST MAY BE RE-ENTERED'
[38]  'WHEN THE SEARCH HAS BEEN COMPLETED, THE DOCUMENT NU
      MBERS'
[39]  '     MAY BE LISTED BY TYPING ''YES'' TO ''OUTPUT DO
      C NOS?'';'
[40]  '     ELSE, ''NO'''
[41]  'THEN ''MORE REQUESTS?'' IS TYPED; IF THE SAME REQUE
      ST IS TO BE'
[42]  '     EXPANDED DIFFERENTLY,'
[43]  'TYPE ''SAME''; ELSE, ''NO'' OR ''END'' '
[44]  →0
[45]  XPLAN3:'
      THE FORMAT OF THE INPUT TO THE ''PRINT'' SUBSYSTEM I
      S AS'
[46]  '     FOLLOWS'
[47]  'DOC NO, OPTION 1 (, OPTION 2) (, OPTION 3) ...'
[48]  'DOC NO IS THE DOCUMENT NUMBER CONCERNED'
[49]  'SIX OPTIONS ARE AVAILABLE
      '
[50]  'TITLE (TI)'
[51]  '     THE TITLE IS LISTED'
[52]  'AUTHOR (AU)'
[53]  '     THE AUTHOR IS LISTED'
[54]  'JOURNAL (JO)'
[55]  '     THE SOURCE JOURNAL IS LISTED'
[56]  'XTERMS (XT)'
[57]  '     THE INDEX TERMS AND THE PERIOD THE DOCUMENT DE
      ALS WITH'
[58]  '     ARE LISTED'
[59]  'ABSTRACT (AB)'
[60]  '     THE ABSTRACT IS LISTED'
[61]  'ALL (AL)'
[62]  '     ALL THE ABOVE OPTIONS ARE LISTED
      '
[63]  'TO EXIT FROM THE ROUTINE, TYPE ''END'' '
[64]  →0
[65]  XPLAN4:'
      INITIALLY, THE CURRENT DOCUMENT NUMBER IS TYPED.  TH
      EN:'
[66]  '''TITLE'' IS TYPED, AND THE USER INPUTS THE TITLE'
[67]  '''AUTHOR'' IS TYPED, AND THE USER INPUTS THE AUTHOR'
[68]  '''JOURNAL'' IS TYPED, AND THE USER INPUTS THE JOURN
      AL IN'
[69]  '     THE FORMAT'
[70]  '
          JOURNALNAME, VOL. XX
      '
```

```
[71]  '      THE VOLUME NUMBER MAY BE OMITTED'
[72]  '''XTERMS'' IS TYPED, AND THE USER INPUTS THE INDEX
      TERMS,'
[73]  '      ONE AT A TIME.  TO TERMINIATE, TYPE ''END'''
[74]  '''YEARS'' IS TYPED, AND THE USER INPUTS THE YEARS I
      N ONE OF'
[75]  '      THE FOLLOWING FORMATS:
      '
[76]  '      1921'
[77]  '      1921,1923'
[78]  '      1921,1923,1928'
[79]  '      1921-1928
      '
[80]  '''ABSTRACT'' IS TYPED, AND THE USER INPUTS THE ABST
      RACT

      '
[81]  '''MORE?'' IS TYPED'
[82]  '      IF THE RESPONSE IS ''NO'', NOTHING IS STORED U
      NDER'
[83]  '      THAT DOCUMENT NUMBER, AND CONTROL IS RETURNED'
[84]  '      TO SARACNL'
[85]  '      IF ''YES'', THE NEXT DOCUMENT NUMBER IS TYPED'
    ∇
```

APPENDIX C

EXAMPLES OF USE OF SARA

Example 1 (STORE routine)

```
     SARACNL
FULL EXPLANATION? YES OR NO
Y

YOU ARE UNDER THE CONTROL OF THE MAINLINE ROUTINE
YOU HAVE 3 ROUTINES AVAILABLE FOR PROCESSING
IN ORDER TO ENTER THE REQUIRED ROUTINE, TYPE ITS NAME AFTER
     'GO' IS TYPED
IN ORDER TO RETURN CONTROL TO THIS ROUTINE, TYPE 'END' AT
     ANY INPUT TIME

FIND
     ACCEPTS INPUT REQUESTS AND RETURNS DOCUMENT NUMBERS
     SATISFYING THE REQUEST
PRINT
     LISTS ANY PART OF THE DOCUMENT WITH THE GIVEN DOCUMENT
     NUMBER
STORE
     ALLOWS STORAGE OF DOCUMENTS
END
     RETURNS CONTROL TO THE APL SYSTEM

GO
STORE
FULL EXPLANATION?
YES

INITIALLY, THE CURRENT DOCUMENT NUMBER IS LISTED AND 'MORE?'
     IS TYPED
IF THE RESPONSE IS 'NO', NOTHING IS STORED UNDER THAT
     DOCUMENT NUMBER, AND CONTROL IS RETURNED TO SARACNL
IF 'YES' IS TYPED, WE CONTINUE


'TITLE' IS TYPED, AND THE USER INPUTS THE TITLE
'AUTHOR' IS TYPED, AND THE USER INPUTS THE AUTHOR
'JOURNAL' IS TYPED, AND THE USER INPUTS THE JOURNAL IN
     THE FORMAT

     JOURNALNAME, VOL. XX

     THE VOLUME NUMBER MAY BE OMITTED
'XTERMS' IS TYPED, AND THE USER INPUTS THE INDEX TERMS,
     ONE AT A TIME.  TO TERMINIATE, TYPE 'END'
```

'YEARS' IS TYPED, AND THE USER INPUTS THE YEARS IN ONE OF THE FOLLOWING FORMATS:

```
    1921
    1921,1923
    1921,1923,1928
    1921-1928
```

'ABSTRACT' IS TYPED, AND THE USER INPUTS THE ABSTRACT

THE NEXT DOCUMENT NUMBER IS THEN TYPED

```
DOC NO 5
TITLE
A UNIVERSITY IN TROUBLE

AUTHOR
THOMPSON-WP

JOURNAL
SASKHIST

XTERMS
EDU
UNIVERSITY
INVALID XTERM - UNIVERSITY
UNIVERSITIES
UNIVERSITY-OF-SASKATCHEWAN
END
YEARS
1919

ABSTRACT


MORE?
NO
GO
END
ALL DONE
```

Example 2 (FIND routine)

```
      SARACNL
FULL EXPLANATION? YES OR NO
NO
GO
FIND
FIND FULL EXPLANATION? YES OR NO
Y

IF WEIGHTS ARE TO BE ATTACHED TO EACH INDEX TERM, TYPE 'YES'
     WHEN REQUESTED (EQUAL WEIGHTS?);
     ELSE,'NO'
THEN, TYPE THE REQUEST IN STANDARD BOOLEAN FORM
WHEN 'EXPANSION?' IS REQUESTED, ONE OF FOUR WORDS (OR THEIR
     ABBREVIATIONS AS IN THE FOLLOWING PARENTHESES)
     IS EXPECTED

NONE (NON)
     NO EXPANSION OF THE REQUEST
GENERAL (GEN)
     INCLUDE GENERAL TERMS IN AN OR RELATIONSHIP
SPECIFIC (SPE)
     INCLUDE SPECIFIC TERMS IN AN OR RELATIONSHIP
RELATED (REL)
     INCLUDE RELATED TERMS IN AN OR RELATIONSHIP

WHEN REQUESTED 'START SCAN?' TYPE 'YES' OR 'NO'.  IF
     'YES', A SEARCH OF ALL DOCUMENTS IS STARTED; IF 'NO',
     A REQUEST MAY BE RE-ENTERED
WHEN THE SEARCH HAS BEEN COMPLETED, THE DOCUMENT NUMBERS
     MAY BE LISTED BY TYPING 'YES' TO 'OUTPUT DOC NOS?';
     ELSE, 'NO'
THEN 'MORE REQUESTS?' IS TYPED; IF THE SAME REQUEST IS TO BE
     EXPANDED DIFFERENTLY, TYPE 'SAME';
     ELSE, 'NO','YES', OR 'END'

EQUAL WEIGHTS?
Y
(PARTIES∨ELECTIONS)∧SASKATCHEWAN∧(PER≥1905)∧(PER≤1930)
EXPANSION?
NONE
START SCAN?
Y
COUNT = 1
OUTPUT DOC NOS?
```

```
Y

   1  13
MORE REQUESTS?
SAME
EXPANSION?
GENERAL
START SCAN?
Y
COUNT = 6
OUTPUT DOC NOS?
Y

   1  13
   2  14
   3  11
   4  37
   5  19
MORE?
Y

   6  17
MORE REQUESTS?
Y
EQUAL WEIGHTS?
NO
FAM∧SASKATCHEWAN∧(PER≥1860)
WEIGHT? - FAM
12
WEIGHT? - SASKATCHEWAN
25
WEIGHT? - PER
2
EXPANSION?
NONE
START SCAN?
Y
COUNT = 1
OUTPUT DOC NOS?
Y

  1  4
MORE REQUESTS?
SAME
EXPANSION?
RELATED
START SCAN?
Y
COUNT = 3
OUTPUT DOC NOS?
```

```
Y

   1  11
   2   6
   3   4
MORE REQUESTS?
NO
GO

FIND NO
EQUAL WEIGHTS?
Y
PRESBYTERIAN∧(PER≥1850)∧(PER≤1920)∧(MISSIONS∨MISSIONARIES)∧
SASKHIST
EXPANSION?
NONE
START SCAN?
Y
COUNT = 1
OUTPUT DOC NOS?
N
MORE REQUESTS?
SAME
EXPANSION?
GENERAL
START SCAN?
Y
COUNT = 7
OUTPUT DOC NOS?
Y

   1  10
   2   3
   3   1
   4   4
   5   6
MORE?
Y

   6  40
   7  31
MORE REQUESTS?
NO
GO
```

```
FIND NO
EQUAL WEIGHTS?
NO
TRAVEL∧(EXPLORERS∨SASKATCHEWAN∨ALBERTA)∧SASKHIST
WEIGHT? - TRAVEL
15
WEIGHT? - EXPLORERS
20
WEIGHT? - SASKATCHEWAN
10
WEIGHT? - ALBERTA
25
WEIGHT? - SASKHIST
5
EXPANSION?
NONE
START SCAN?
Y
COUNT = 0
MORE REQUESTS?
SAME
EXPANSION?
GENERAL
START SCAN?
YES
COUNT = 3
OUTPUT DOC NOS?
Y

   1  42
   2  41
   3  38
MORE REQUESTS?
NO
GO
```

```
FIND NO
EQUAL WEIGHTS?
Y
MACKAY-JA∧REL∧(PER≤1907)∧(~GEO)
EXPANSION?
NONE
START SCAN?
Y
COUNT = 2
OUTPUT DOC NOS?
Y

  1   3
  2   1
MORE REQUESTS?
SAME
EXPANSION?
SPECIFIC
START SCAN?
Y
COUNT = 2
OUTPUT DOC NOS?
Y

  1   3
  2   1
MORE REQUESTS?
NO
GO
```

```
FIND NO
EQUAL WEIGHTS?
NO
LOCAL-GOVERNMENT∧(NORTHWEST-TERRITORIES∨PRINCE-ALBERT
∨BALCARRES)∧(PER≤1905)
WEIGHT? - LOCAL-GOVERNMENT
4
WEIGHT? - NORTHWEST-TERRITORIES
5
WEIGHT? - PRINCE-ALBERT
7
WEIGHT? - BALCARRES
2
WEIGHT? - PER
1
EXPANSION?
NONE
START SCAN?
YES
COUNT = 2
OUTPUT DOC NOS?
Y

   1  21
   2  20
MORE REQUESTS?
SAME
EXPANSION?
RELATED
START SCAN?
Y
COUNT = 6
OUTPUT DOC NOS?
Y

   1  10
   2   8
   3  18
   4  21
   5  20
MORE?
Y

   6  24
MORE REQUESTS?
NO
GO
```

Example 3 (PRINT routine)

```
      SARACNL
FULL EXPLANATION? YES OR NO
N
GO
GO
INCORRECT COMMAND
GO
PRINT
FULL EXPLANATION? YES OR NO
Y

THE FORMAT OF THE INPUT TO THE 'PRINT' SUBSYSTEM IS AS FOLLOWS:
     DOC NO, OPTION 1 (, OPTION 2) (, OPTION 3) ...
DOC NO IS THE DOCUMENT NUMBER CONCERNED
SIX OPTIONS ARE AVAILABLE

TITLE (TI)
     THE TITLE IS LISTED
AUTHOR (AU)
     THE AUTHOR IS LISTED
JOURNAL (JO)
     THE SOURCE JOURNAL IS LISTED
XTERMS (XT)
     THE INDEX TERMS AND THE PERIOD THE DOCUMENT DEALS WITH ARE
      LISTED
ABSTRACT (AB)
     THE ABSTRACT IS LISTED
ALL (AL)
     ALL THE ABOVE OPTIONS ARE LISTED

TO EXIT FROM THE ROUTINE, TYPE 'END'


 PRINT WHAT?
16,TITLE,AUTHOR
DOMINION GOVERNMENT AID TO THE DAIRY INDUSTRY IN WESTERN CANADA,
       1890-1906
CHURCH-GC

 PRINT WHAT?
11,XT,AB
YEARS - 1860  0  1919
 BIO POL BROWN-GW SASKATCHEWAN NORTHWEST-TERRITORIES POLITICIANS

NONE

 PRINT WHAT?
```

```
23,ALL
QUIET EARTH, BIG SKY
STEGNER-W
SASKHIST
YEARS - 1915  0  1919
 SOC EASTEND SASKATCHEWAN PIONEER-LIFE
NONE

 PRINT WHAT?
END
GO
END
ALL DONE
```